TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

DISSERTATION

Large Data Scalability in Interactive Visual Analysis

ausgeführt zum Zwecke der Erlangung des akademischen Grades
eines Doktors der technischen Wissenschaften

unter der Leitung von

Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Eduard Gröller,
Institut E186 für Computergraphik und Algorithmen,

eingereicht an der Technischen Universität Wien,
Fakultät für Informatik,

von

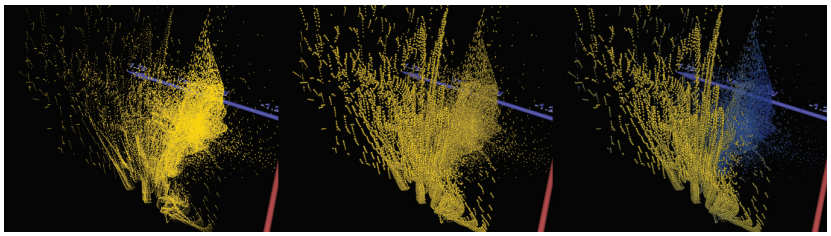
Dipl.-Ing. Harald Piringer,

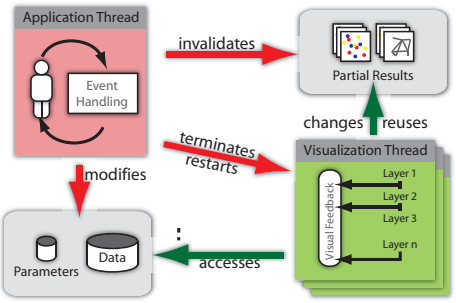
Matrikelnummer 9826148,
Flötzersteig 284/B4,
A-1140 Wien

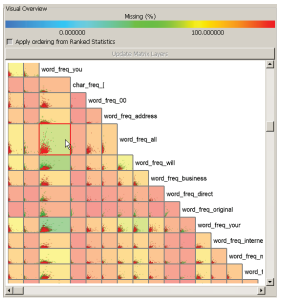
Wien, im Mai 2011

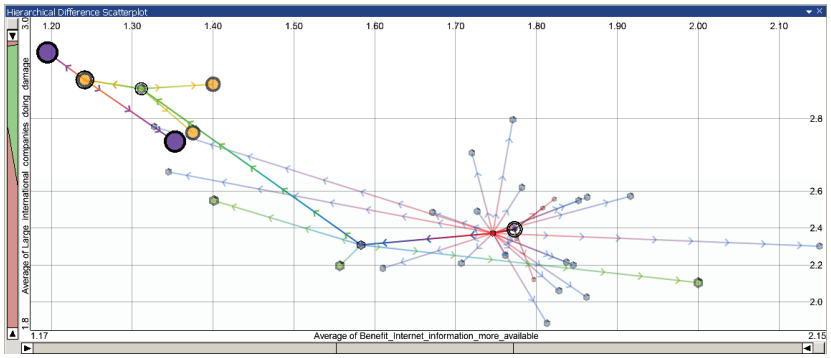
Large Data Scalability in Visual Analysis

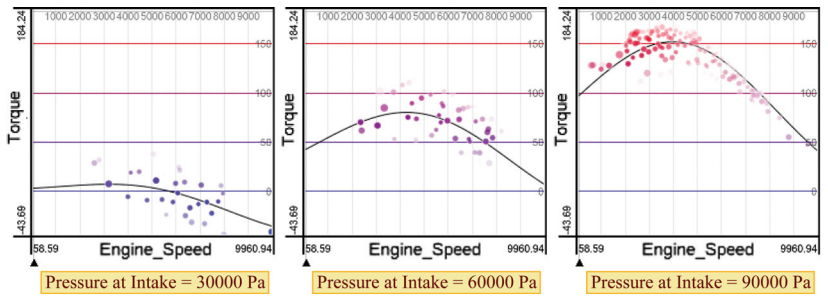
Harald Piringer, PhD thesis











mailto:piringer@vrvis.at
http://www.vrvis.at/forschung/visual-analysis

Abstract

In many areas of science and industry, the amount of data is growing fast and often already exceeds the ability to evaluate it. On the other hand, the unprecedented amount of available data bears an enormous potential for supporting decision-making. Turning data into comprehensible knowledge is thus a key challenge of the 21st century.

The power of the human visual system makes visualization an appropriate method to comprehend large data. In particular interactive visualization enables a discourse between the human brain and the data that can transform a cognitive problem to a perceptual one. However, the visual analysis of large and complex datasets involves both visual and computational challenges. Visual limits involve perceptual and cognitive limitations of the user and restrictions of the display devices while computational limits are related to the computational complexity of the involved algorithms.

The goal of this thesis is to advance the state of the art in visual analysis with respect to the scalability to large datasets. Due to the multifaceted nature of scalability, the contributions span a broad range to enhance computational scalability, to improve the visual scalability of selected visualization approaches, and to support an analysis of high-dimensional data.

Concerning computational scalability, this thesis describes a generic architecture to facilitate the development of highly interactive visual analysis tools using multi-threading. The architecture builds on the separation of the main application thread and dedicated visualization threads, which can be cancelled early due to user interaction. A quantitative evaluation shows fast visual feedback during continuous interaction even for millions of entries.

Two variants of scatterplots address the visual scalability of different types of data and tasks. For continuous data, a combination of 2D and 3D scatterplots intends to combine the advantages of 2D interaction and 3D visualization. Several extensions improve the depth perception in 3D and address the problem of unrecognizable point densities in both 2D and 3D. For partly categorical data, the thesis contributes Hierarchical Difference Scatterplots to relate multiple hierarchy levels and to explicitly visualize differences between them in the context of the absolute position of pivoted values.

While comparisons in Hierarchical Difference Scatterplots are only qualitative, this thesis also contributes an approach for quantifying subsets of the data by means of statistical moments for a potentially large number of dimensions. This approach has proven useful as an initial overview as well as for a quantitative comparison of local features like clusters.

As an important application of visual analysis, the validation of regression models also involves the scalability to multi-dimensional data. This thesis describes a design study of an approach called HyperMoVal for this task. The key idea is to visually relate n-dimensional scalar functions to known validation data within a combined visualization. The integration with other multivariate views is a step towards a user-centric workflow for model building.

Being the result of collaboration with experts in engine design, HyperMoVal demonstrates how visual analysis is suitable to significantly improve real-world tasks. Positive user feed-

back suggests a high impact of the contributions of this thesis also outside the visualization research community. Moreover, most contributions of this thesis have been combined in a commercially distributed software framework for engineering applications that will hopefully raise the awareness and promote the use of visual analysis in multiple application domains.

Kurzfassung

In vielen Bereichen von Wissenschaft und Industrie wachsen die Datenmengen so rasch, dass sie oftmals nicht mehr ausgewertet werden können. Andererseits birgt die noch nie dagewesene Verfügbarkeit von Daten ein enormes Potential zur Unterstützung von Entscheidungsfindungen. Die datenbasierte Wissensgewinnung ist somit eine zentrale Herausforderung des 21. Jahrhunderts.

Dank der Leistungsfähigkeit des menschlichen Sehapparates ist Visualisierung ein geeignetes Mittel um große Datenmengen zu verstehen. Speziell interaktive Visualisierungen ermöglichen einen Diskurs mit Daten, der es erlaubt, kognitive Aufgaben durch visuelle Wahrnehmung zu lösen. Allerdings umfasst die visuelle Analyse großer und komplexer Daten Herausforderungen sowohl was die Darstellung, als auch was die Berechnung angeht. Erstere betreffen perzeptuelle und kognitive Grenzen von Benutzern während letztere eng mit der Komplexität der eingesetzten Algorithmik zusammenhängen.

Ziel dieser Dissertation ist die Erweiterung des Stands der Technik im Bereich visueller Analyse bezüglich der Skalierbarkeit für große Datenmengen. Entsprechend der vielen Facetten des Themas Skalierbarkeit spannen die Innovationen dieser Dissertation einen weiten Bogen von berechnungsbezogener Skalierbarkeit über die Verbesserung der visuellen Skalierbarkeit ausgewählter Visualisierungsansätze bis hin zur Unterstützung einer Analyse hochdimensionaler Daten.

Hinsichtlich berechnungsbezogener Skalierbarkeit beschreibt diese Dissertation eine generische Architektur, um den Einsatz von Multithreading bei der Entwicklung interaktiver visueller Analysesysteme zu erleichtern. Kern der Architektur ist die Trennung des Hauptthreads der Applikation von speziellen Visualisierungsthreads sowie deren vorzeitigen Abbruch im Falle von Benutzerinteraktion. Eine quantitative Evaluierung belegt ein rasches visuelles Feedback während kontinuierlichen Interaktionen selbst bei Millionen von Datenwerten.

Zwei Varianten von Punktdiagrammen widmen sich der visuellen Skalierbarkeit verschiedener Arten von Daten und Aufgaben. Im Falle kontinuierlicher Daten beabsichtigt eine Kombination aus 2D und 3D Punktdiagrammen die Vorteile zweidimensionaler Interaktion und dreidimensionaler Visualisierung miteinander zu kombinieren. Diverse Erweiterungen verbessern die Tiefenwahrnehmung in 3D und widmen sich dem Problem einer nicht erkennbaren Datendichte sowohl in 2D als auch in 3D. Für den Fall teilweiser kategorischer Daten beschreibt die Dissertation eine als hierarchische Differenz-Punktdiagramme (Hierarchical Difference Scatterplots) bezeichnete Technik. Zweck ist es, mehrere Hierarchiestufen miteinander in Bezug zu setzen und deren Unterschiede in Bezug auf diverse Aggregate explizit darzustellen.

Während Vergleiche in hierarchischen Differenz-Punktdiagrammen nur qualitativer Natur sind, stellt diese Dissertation auch einen quantitativen Ansatz vor, der darauf beruht, statistische Momente von Teilmengen der Daten für potentiell viele Dimensionen gleichzeitig zu

ermitteln. Anwendungen dieses Ansatzes umfassen sowohl einen Überblick über die Gesamtdaten als auch einen quantitativen Vergleich lokaler Charakteristika wie beispielsweise Cluster.

Die Validierung von Regressionsmodellen ist eine wichtige Anwendung für eine visuelle Analyse, die eine Skalierung hinsichtlich höher dimensionaler Daten erfordert. Für diese Anwendung wird eine Designstudie eines als HyperMoVal bezeichneten Ansatzes beschrieben. Kernidee ist es, n-dimensionale Skalarfunktionen mit bekannten Validierungsdaten in einen gemeinsamen visuellen Bezug zu bringen. Die Integration mit anderen multivariaten Darstellungen stellt dabei einen Schritt in Richtung eines Benutzer-basierten Modellbildungsprozesses dar.

Als Ergebnis einer Zusammenarbeit mit Experten im Bereich Motorenentwicklung zeigt HyperMoVal auch, dass visuelle Analyse geeignet ist, reale Aufgaben erheblich zu erleichtern. Positives Feedback seitens von Benutzern deutet die Bedeutung der Innovationen dieser Dissertation auch außerhalb der Forschungsgemeinde im Bereich Visualisierung an. Umso mehr, als die meisten Innovationen dieser Dissertation in einem gemeinsamen Softwareframework für Ingenieursanwendungen kommerziell vertriebenen werden. Es ist zu hoffen, dass dieses das Bewusstsein um die Möglichkeiten visueller Analyse und deren Anwendung in unterschiedlichen Bereichen steigern wird.

Contents

Related Publications

This thesis is based on the following publications:

Harald Piringer, Robert Kosara, and Helwig Hauser
Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots,
Proceedings of the 2nd International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2004), pp. 49 – 60, 2004.

Harald Piringer, Wolfgang Berger, and Helwig Hauser
Quantifying and Comparing Features in High-Dimensional Datasets,
Proceedings of the 6th International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2008), pp. 240 – 245, 2008.

Harald Piringer, Christian Tominski, Philipp Muigg, and Wolfgang Berger
A Multi-Threading Architecture to Support Interactive Visual Exploration,
IEEE Transactions on Visualization and Computer Graphics, 15(6), pp. 1113 – 1120, 2009.

Harald Piringer, Matthias Buchetics, Helwig Hauser, and Eduard Gröller
Hierarchical Difference Scatterplots - Interactive Visual Analysis of Data Cubes,
SIGKDD Explorations, 11(2), pp. 49 – 58, 2009.

Harald Piringer, Wolfgang Berger, and Jürgen Krasser
HyperMoVal: Interactive Visual Validation of Regression Models for Real-Time Simulation,
Computer Graphics Forum, 29(3), pp. 983 – 992, 2010.

Chapter 1

Introduction and Overview

This chapter introduces the motivations and challenges of a scalable visual data analysis, and it provides an overview of the main contributions of this thesis.

1.1 Motivation

We are drowning in information and starving for knowledge. This quote by Rutherford D. Rodgers summarizes one of the most critical challenges of our time. The ubiquitous use of information technology in most areas of science and industry, the exponential growth in computing power and storage capacity, and improvements in sensors and recording methods have led to the current situation that data is collected and generated at an incredible rate. Examples of collected data per day include 300 million VISA credit card transactions, thousands of complex financial stocks with millions of transactions, and 210 billion emails [146]. Examples of generated data are terabytes of simulation results in engineering, physics, biology, and climate research. Even today, the rate at which data is collected and stored exceeds the human ability to use that data for decisions in many fields. This has been termed "information glut" [212] and "information overload" [146]. The explosion of the digital content [88] suggests that it will become even worse in the future. Without being able to make sense of the contained information, however, data becomes useless.

From a negative point of view, potential consequences of the information overload range from a waste of time and money in case of business intelligence to human casualties in case of security applications. From a positive point of view, the unprecedented amount of available data bears an enormous potential for gaining knowledge and supporting decision-making. As a concrete benefit, having more complete data may reduce uncertainty or facilitate the detection of missing information. Moreover, analyzing large and complex amounts of data is also the key to solving some of the most important challenges of our time. Environmental sustainability, for example, is one of the UN millennium goals [189]. To achieve this goal, engineering the tools of scientific discovery has been ranked among the grand challenges for engineering [195]. A major purpose of such tools will be to support scientists in turning data into comprehensible knowledge.

While there are several methods to analyze data as discussed in the next section, a *visual analysis* has many advantages. The famous proverb "a picture is worth 10.000 words" suggests a close relationship between perception and cognition. Moreover, "seeing" and "understanding" are synonyms in English and "insight" is also related to vision. Humans acquire

more information through vision than through all other senses combined [268]. This makes the human visual system an enormously powerful pattern seeker combining 20 billion neurons as a massively parallel processor with the highest bandwidth channel into human cognitive centers. Furthermore, some researchers emphasize that "the world is its own memory", which means that our ability to think is limited without external representation [197, 268]. All these facts are strong evidence that visualization is indeed an appropriate method to comprehend huge amounts of data.

1.2 From Static Images to Visual Analysis: A Short History of Visualization

The verb *to visualize* has two meanings. "To form a mental image of something" refers to a cognitive, internal aspect whereas "to make something visible to the eye" refers to an external, perceptual role [50]. While these two meanings once more emphasize the relationship between perception and cognition, Keim et al. argue that the most common understanding of visualization has changed over time and now mostly refers to a graphical representation of data or concepts [147]. The goals of visualization have changed in a similar way as its meaning. According to Keim et al., visualization has three major goals:

1. *Presentation* refers to an efficient and effective communication of facts that are fixed a priori.

2. *Confirmatory analysis* can be described as a goal-oriented examination of existing hypotheses with the aim of confirming or rejecting them.

3. *Exploratory analysis* is a typically undirected search for new information like structures and trends without initial hypothesis.

While early uses of visualization – mostly maps – date back to pre-Christian times [268, 10], the by far most common goal was *presentation* until the end of the 20th century. As a rare documented exception, identifying a contaminated well as the cause of a Cholera epidemic in 19th century London was an early example where visualization helped to generate new insights [90] (see Fig. 1.1). In general, however, the limitation to printed graphics restricted visualization to a static means for presenting existing knowledge for a long time.

The rise of *exploratory data analysis* began with the age of computers and improvements in graphical user interfaces. In 1977, a book by John W. Tukey [257] had a major influence on promoting exploratory data analysis in the statistics research community. This was an important step, as data analysis has historically been a mostly statistical issue, and many common types of visualizations like scatterplots or box plots originate from statistics.

Utilizing computers to generate visualizations has emerged as an own research discipline during the last two decades. A milestone of computer-oriented visualization research was the move from static images to *interactive visualization*. Interaction enables a discourse between the human brain and the data that, for example, allows to focus on interesting structures and to rapidly try many what-if scenarios in an ad-hoc fashion. While the role of interaction is still a subject of ongoing research [286, 167], interactions like filtering data, changing visualization parameters at run-time, and linking multiple visualizations have soon become a standard (see also chapter 2).

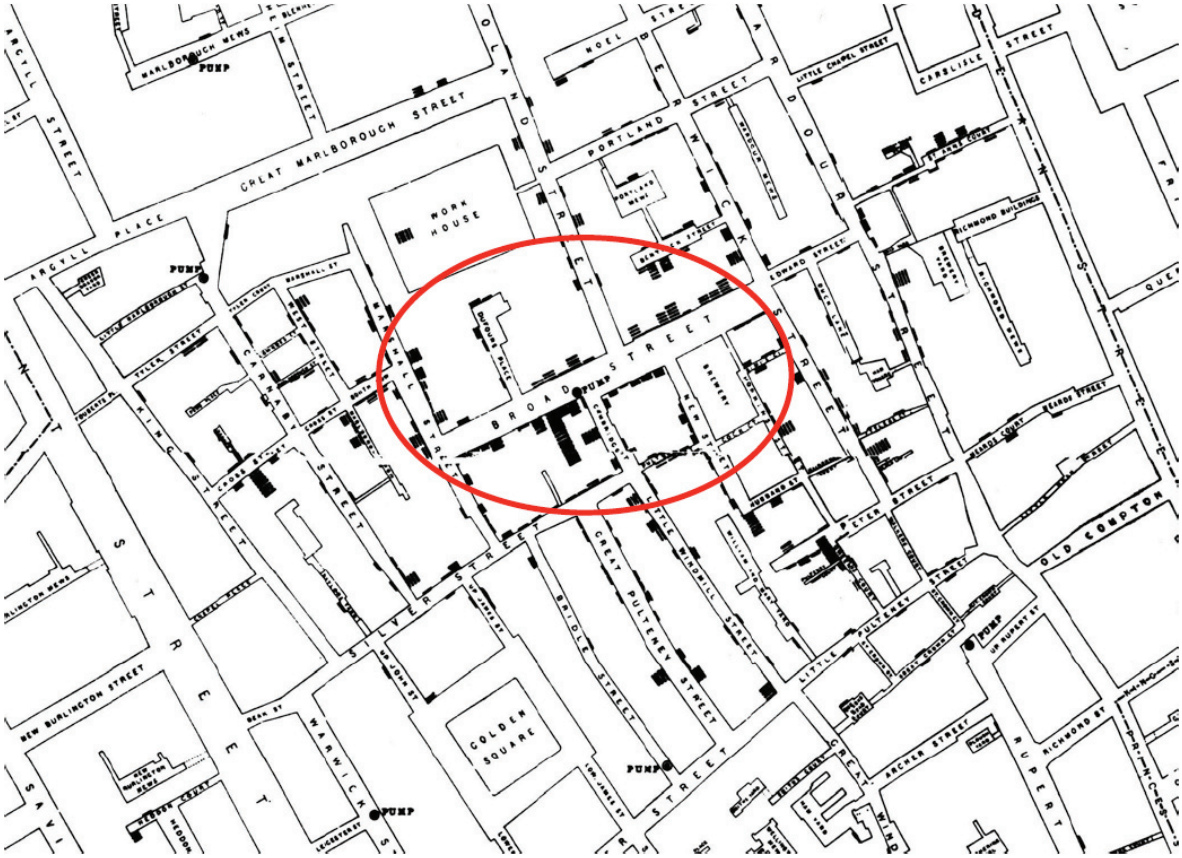


Figure 1.1: In 1844, this visualization led Dr. John Snow to the discovery that the death cases (indicated by bars) are clustered around the encircled water pump [90].

Research in visualization is usually broadly classified into scientific and information visualization. *Scientific visualization* [101] comprises methods where an inherent mapping between the data and coordinates in a virtual environment exists. Typical applications are volume rendering for 3D scalar fields like medical body scans as well as flow visualization of 3D or 4D vector fields as, for example, obtained from computational fluid dynamics. In contrast, *information visualization* is defined as the use of interactive visual representations of abstract data to amplify cognition [34]. In this definition, abstract data refers to a lack of explicit spatial references for parts or all of the data. Examples include data that is categorical, high-dimensional, textual, hierarchical, or relational (see Fig. 1.2). A key aspect of information visualization is to find understandable visual metaphors in concert with intuitive interaction techniques. While a classification in scientific and information visualization is reasonable on a technique-level, a holistic analysis of real-world data often requires a combination of different visualization methods [52].

Scientfic and information visualization also share many concrete *benefits* with respect to exploratory data analysis. A major advantage of interactive visualization in general is the ability to transform a cognitive problem to a perceptual one. Interactive visualization facilitates hypothesis formation by fostering a detection of both large-scale and small-scale features in potentially massive data that were not anticipated, e.g., artefacts like wrong or missing values [268]. Despite these advantages, van Wijk warns that visualization is not

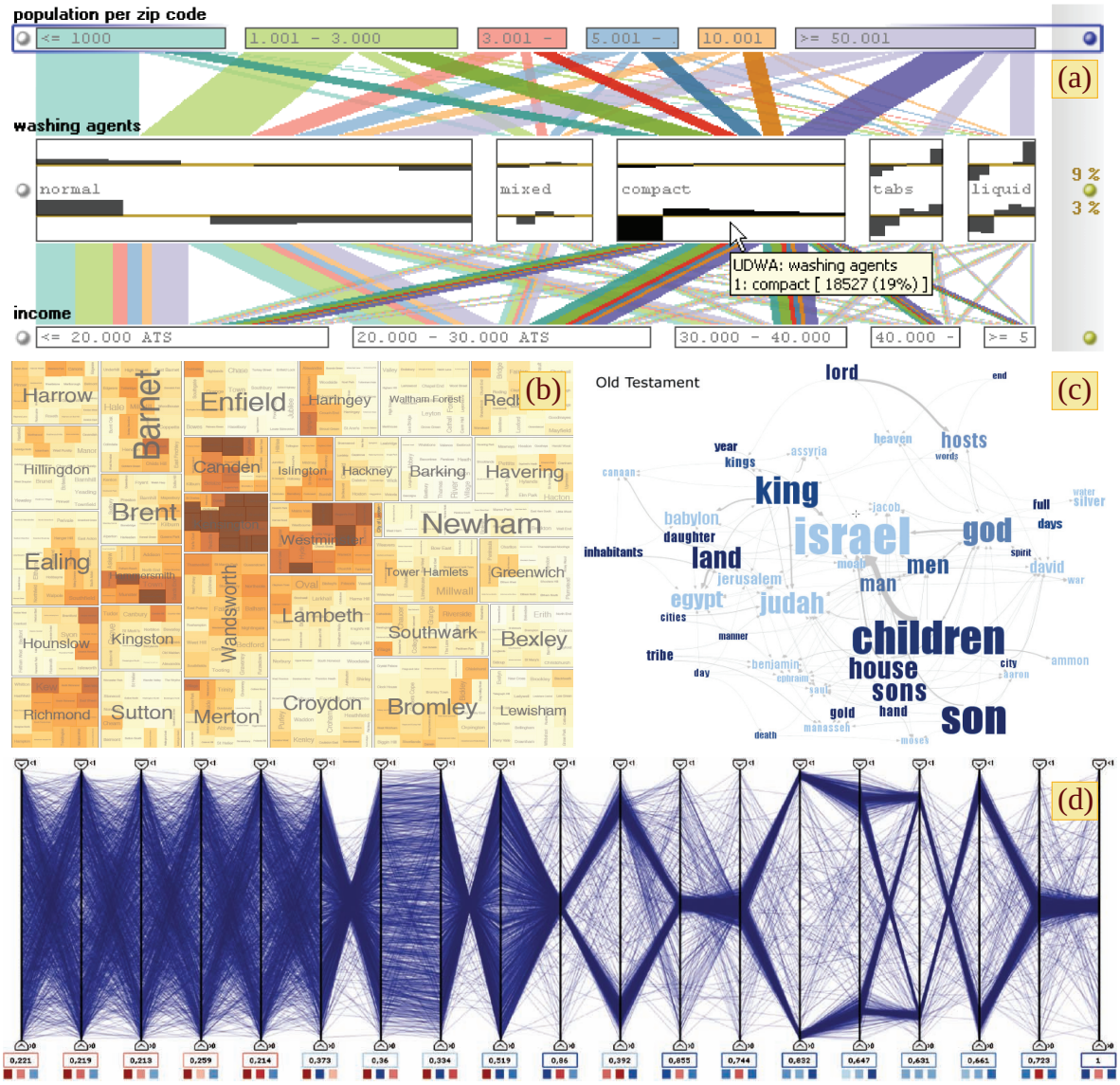


Figure 1.2: Examples of information visualization. (a) Parallel sets showing categorical data [154], (b) a tree map for hierarchical data [231], (c) a graph indicating relations between words in an unstructured text [259], (d) parallel coordinates of 19 dimensions [136].

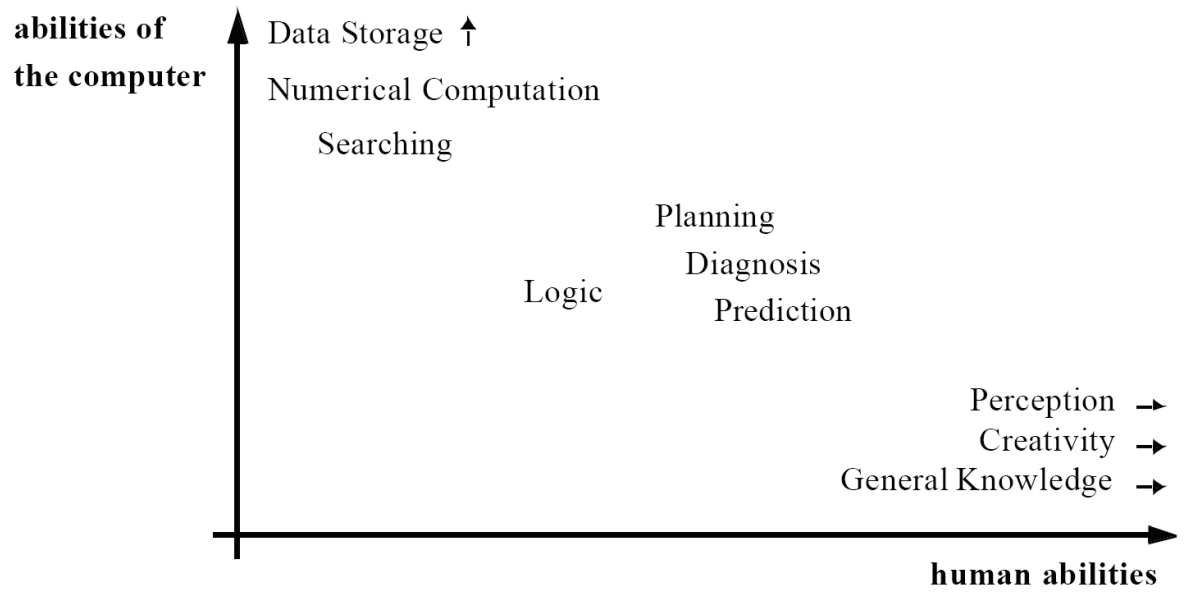


Figure 1.3: Comparing the complementary abilities of humans and computers [145].

"good" by definition, but involves costs for development, training, and data preparation as well as perception and exploration costs [260]. Many of these costs are caused by the inherent involvement of human users. However, human involvement is neither necessary nor desirable for all types of tasks. Moreover, most visualization techniques do not scale to truly high-dimensional data, and visual results are often only qualitative which is not fully sufficient for many tasks.

Automated approaches to data analysis (partly) avoid costs due to human involvement. Most approaches yield precise values as results and many approaches scale well to high-dimensional data. Statistical learning and data mining, for example, have long been applied to make predictions without human interaction in fields like business intelligence, pattern recognition, and science [19]. There are numerous techniques for automated classification, prediction, and clustering [106]. Therefore, using automated techniques is typically preferable if the properties of the data are known and the goals of the analysis can precisely be specified a priori. However, for increasingly complex problems, purely automated analysis is often insufficient. The reasons range from a lack of understanding of the results of automated methods to the inability to include human knowledge in case of conflicting, heterogeneous, or messy data.

Interactive visualization and automated data analysis are thus two different approaches with similar goals that have *complementary advantages and disadvantages* [20]. This corresponds to the fact that human abilities like perception, creativity, and general knowledge are complementary to the strengths of computers like processing power and storage capacity [145] (see Fig. 1.3). A next major step of visualization is consequently to strive for a tight integration of interactive visual and automated data analysis. In 2005, the term *visual analytics* was established for the respective research direction. As an early definition, visual analytics was defined as "the science of analytical reasoning facilitated by interactive visual interfaces" [247]. More recently, the definition has been concretized: "Visual analytics combines

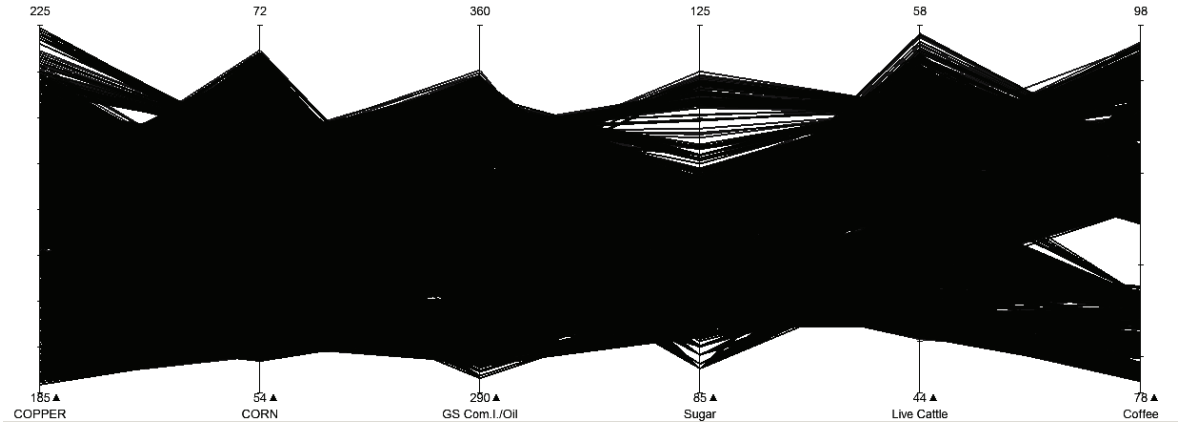


Figure 1.4: Parallel coordinates of approximately 30.000 data entries as an example of clutter.

automated analysis techniques with interactive visualizations for an effective understanding, reasoning and decision making on the basis of very large and complex datasets." [146]. An important aspect of this latter definition is the emphasis on scale and complexity, which is the topic of this thesis.

1.3 Scalability in Visual Data Analysis

The increasing size and complexity of datasets is a key motivation for interactive visual approaches. *Scalability* is thus a core topic of visual analysis. The importance of scalability is stressed by the fact that it has been approved as the Priority Program *Scalable Visual Analytics: Interactive Visual Analysis Systems of Complex Information Spaces* (SPP 1335) by the Senate of the Deutsche Forschungsgemeinschaft. Scalability has also been described as a grand challenge in the research agenda of visual analytics [247].

1.3.1 Large Data Scalability

While scalability is a multifaceted problem as will be discussed in section 1.3.2, the sheer *size* of a dataset is a major aspect. A visual analysis becomes increasingly challenging with a growing amount of data. Dix and Ellis explain the problems of large data by visual and computational limits [62].

Visual limits are caused by perceptual and cognitive limitations of the user as well as hardware limitations of the display device. The main challenge is to visually represent a very large number of data elements in a much smaller number of visual display elements [212]. The human perception of patterns in visual displays adheres to the Gestalt laws [268]. Continuity and closure, for example, are often required to convey certain properties of the data. Cluttering a visualization by too many data elements affects the perception in a negative way. In scatterplots or parallel coordinates, for example, overplotting makes it impossible to judge the true distribution of the data. In an extreme case, the visualization becomes a single uniform blob (see Fig. 1.4 for an example). Using visual attributes like color, shape, or size to convey additional data attributes can make the problem even worse. Colors of closely spaced pixels will be merged by the eye, and glyphs are likely to overlap and obscure each other [62].

The capability of a visualization to effectively display large datasets in terms of either the number or the dimension of individual data elements is known as its *visual scalability* [61]. Eick and Carr identified six factors affecting visual scalability, i.e., human perception, monitor resolution, visual metaphors, interactivity, data structures and algorithms, and computational infrastructure. Besides human perception which is inherently given, the other five factors explain a great diversity of different visualization approaches with respect to visual scalability. In particular, visualization research has designed a huge variety of visual metaphors and interaction concepts to improve the visual scalability (see chapter 2).

However, visualization techniques have a different visual scalability with respect to the *number of displayed data elements* and the *number of concurrently shown data dimensions*. Most visualization techniques have no inherent limit with regards to the number of displayed data elements and a variety of clutter reduction techniques exist to overcome practical limitations [64]. On the other hand, most techniques are inherently limited with respect to the number of dimensions which can simultaneously be displayed (e.g., scatterplots). For other techniques like parallel coordinates or scatterplot matrices, the practical limit for the number of dimensions is magnitudes smaller than the one for data elements. As a consequence, an analysis of truly high-dimensional data is challenging. The user typically has to pre-select the displayed dimensions, which may become a difficult task without a-priori knowledge or dedicated support.

Computational limits of visualizations are closely related to the computational complexity of the involved algorithms, which is a core topic of computer science [77]. In case of algorithms with a quadratic effort, for example, visualizing hundred items may be interactive while visualizing millions of items may take hours for each update. Approaches to overcome computational limits can be classified as hardware-oriented or software-oriented. As surveyed in chapter 2, hardware-oriented approaches involve parallelization and distributed data storage. Software-oriented approaches include the removal of data using sampling or filtering as well as data aggregation as for multi-resolution approaches.

Visual and computational limits are not independent from each other. Many approaches address both limits simultaneously. Sampling a large number of data items, for example, reduces the computational effort and enhances the visual scalability at the same time [62]. In general, overcoming computational limits is often a necessary prerequisite to achieve visual scalability, because interactivity is an important factor of visual scalability as discussed above. This fact is becoming a challenging issue in the context of an increasingly tight integration of automated and visual analysis, as most automated approaches have not been designed with interaction in mind.

1.3.2 Other Scalability Issues

The contributions of this thesis focus on scalability with respect to large datasets (see section 1.4). The size of the analyzed data, however, is just one aspect of scalability. A recent survey on scale and complexity in visual analytics considers *five major issues of scalability* [212], i.e., information scalability, visual scalability, display scalability, human scalability, and computational scalability. While visual scalability has been discussed in section 1.3.1, this section briefly covers the other types of scalability.

Robertson et al. define *information scalability* as the ability to offer simple visualizations of the right subset of a massive stream of data [212]. Information scalability can thus be seen in a more general sense than dealing with data that is solely large, but it also includes

the rate of change for dynamic data and the facility to scale the presentation to a certain audience. Though not mentioned by Robertson et al., one could also think of other aspects of information scalability. The ability to scale to a large number of different data sources will become an increasingly important issue for visual analysis systems. As a related topic, information scalability also involves the ability to handle data that is heterogeneous in multiple ways [140].

Display scalability refers to the ability of a visualization to be effective from personal digital assistants to wall-sized displays. Today, most visualization systems are designed for desktop displays and are neither suitable for the limited resolution of small screens, nor make effective use of very large screens.

Human scalability refers to the number of humans involved in analytical problem-solving activities. The goal is to achieve a graceful scaling from a single user to a collaborative environment.

Computational scalability stresses the fact that most algorithms do not automatically become faster with an increasingly parallel computing infrastructure. On the contrary, within 15 years, systems for exascale computing are expected to have several million cores, which will require a fundamental paradigm shift for algorithms and visualization approaches. Moreover, for all types of computers, the number of cores is expected to grow significantly faster than the total amount of memory or disk space. This means that the memory resources per process will actually decrease.

Concluding, scalability comes in many different ways. Each scalability issue is a challenge and a research topic in its own right. Interpreting scalability in a broad sense, it has been a driving motivation behind most research in visual data analysis in the past few years.

1.4 Contributions

The goal of this thesis is to advance the state of the art in visual analysis with respect to the scalability to large datasets. Corresponding to the multifaceted nature of scalability, the contributions span a broad range and address different selected topics of interactive large data visualization. In particular, the contributions intend to overcome computational limits, visual limits with regards to the number of data entries, and limits with regards to the dimensionality of particular tasks. The subsequent sections briefly motivate and summarize each contribution. It is also discussed, how each contribution relates to the overall topic of large data scalability in visual analysis.

1.4.1 A Multi-Threading Visualization Architecture

During continuous user interaction, it is hard to provide rich visual feedback at interactive rates for datasets containing millions of entries. Many approaches provide a fixed amount of feedback during a continuous user interaction, which either leaves time unused or may severely degrade the responsiveness of the application.

This thesis contributes a *generic architecture* that ensures the responsiveness of the application even when dealing with large data and that is applicable to many types of visualizations. The 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, the architecture facilitates generating previews incrementally to provide

rich visual feedback quickly. To help avoiding common pitfalls of multi-threading, synchronization and communication are discussed in detail. Explicitly denoted design choices enable to control trade-offs. A quantitative evaluation based on the system Visplore shows fast visual feedback during continuous interaction even for millions of entries. Further instantiations of the architecture in additional tools demonstrate the general applicability.

The multi-threading architecture relates to scalability issues in multiple ways. First, it supports information scalability as it enables systems to remain responsive while scaling to datasets with several million data items. Second, it increases computational scalability by utilizing commonplace multi-core technology. Third, the architecture improves visual scalability in so far as it guarantees visual feedback as quickly as possible, i.e., it keeps the latency between interaction and visual feedback below 100 ms [226]. Fourth, it scales with regard to multiple views.

Being instantiated in the systems Visplore and SimVis, the multi-threading architecture also relates to all other contributions of this thesis, which are implemented in either of these systems and adhere to the conceptual paradigms.

1.4.2 Focus+Context Visualization with 2D/3D Scatterplots

Scatterplots in 2D and 3D are very useful tools, but also suffer from a number of problems. Overplotting hides the true number of points that are displayed, and showing point clouds in 3D is problematic both in terms of perception and interaction.

This thesis contributes a *combination of 2D and 3D scatterplots*, together with some extensions to both, to overcome these problems. By linking 2D and 3D views, it is possible to interact in 2D and to get feedback in 3D. Several depth cues enhance that feedback in order to provide a better depth impression. Histograms in 2D and 3D show additional information about point densities, and additional context information can be displayed. An example application from the field of computational fluid dynamics demonstrates the usefulness of the technique.

The proposed approach relates to scalability mostly with respect to visual scalability. Using color, halos, and point size as depth cues significantly improves the perception of large 3D point clouds of continuous data attributes. Due to binning, the density information provided by histograms scales well to millions of data entries. Zooming into the data also enhances the scalability by means of interaction while different projection techniques ensure that the spatial context is not lost in this case.

1.4.3 Hierarchical Difference Scatterplots

Data cubes as employed by On-Line Analytical Processing (OLAP) play a key role in many application domains. The analysis typically involves a comparison of categories from different hierarchy levels with respect to size and pivoted values. Most existing visualization methods for pivoted values, however, are limited to single hierarchy levels. On the other hand, most tree visualizations convey the topology of a hierarchy but disregard multivariate attributes.

This thesis contributes an approach called *Hierarchical Difference Scatterplots* (HDS). HDS allow for relating multiple hierarchy levels and explicitly visualize differences between them in the context of the absolute position of pivoted values. Additional contributions involve a discussion concerning a tight coupling of HDS to other types of tree visualizations,

the integration in a setup of multiple linked multivariate views, and an analysis of social survey data in collaboration with a domain expert as evaluation of the approach.

HDS relate to the visual scalability when analyzing data cubes. The overall goal was to combine the visual scalability of overview summaries with a refined degree of detail for selected parts of the data. As the main consideration of the visual encoding of HDS, representing different hierarchy levels in the same visualization makes comparisons much more intuitive and precise than relying on comparisons across multiple views, which is the standard today (e.g., as provided by Tableau [235]). Due to aggregation, HDS scale to datasets with millions of underlying data records. Interaction concepts enable to focus on particular parts of the hierarchy, e.g., comparisons along the hierarchy or across one hierarchy level. In this respect, the approach scales to comparing more than ten hierarchy levels at the same time.

1.4.4 Quantifying and Comparing Features in High-Dimensional Datasets

Linking and brushing is a proven approach to analyze multi-dimensional datasets in the context of multiple coordinated views. Nevertheless, most visualization techniques only offer qualitative visual results for brushed subsets of the data. Many user tasks, however, also require precise quantitative results as, for example, offered by statistical analysis.

Motivated by the Rank-by-Feature Framework [222], this thesis contributes a *joint visual and statistical approach* for guiding the user through a high-dimensional dataset by ranking dimensions (1D case) and pairs of dimensions (2D case) according to statistical summaries. While the original Rank-by-Feature Framework is limited to global features, the most important novelty of the proposed approach is the concept to consider local features, i.e., data subsets defined by brushing in linked views. The ability to compare subsets to other subsets and subsets to the whole dataset in the context of a large number of dimensions significantly extends the benefits of the approach especially in later stages of an exploratory data analysis. A case study illustrates the workflow by analyzing counts of keywords for classifying e-mails as spam or no-spam.

As the most important aspect with regards to scalability, the approach scales to high-dimensional datasets. In particular, ranking different measures of interest enables to quickly identify the most relevant dimensions for hundreds of dimensions in the 1D case. Approximately 35 to 40 dimensions can reasonably be handled in the 2D case. With regards to the number of data items, the statistical summaries have no inherent limitations. However, the preview visualizations in the 2D case suffer from overplotting in case of many data items.

1.4.5 Interactive Visual Validation of Regression Models

During the development of car engines, regression models that are based on machine learning techniques are increasingly important for tasks which require a prediction of results in real-time. While the validation of a model is a key part of its identification process, existing computation- or visualization-based techniques do not adequately support all aspects of model validation.

This thesis contributes an interactive approach called *HyperMoVal* that is designed to support multiple tasks related to model validation: 1) comparing known and predicted results, 2) analyzing regions with a bad fit, 3) assessing the physical plausibility of models also outside regions covered by validation data, and 4) comparing multiple models. The key idea is to visually relate one or more n-dimensional scalar functions to known validation data within

a combined visualization. HyperMoVal lays out multiple 2D and 3D sub-projections of the n-dimensional function space around a focal point. As a related contribution, concepts for linking HyperMoVal to other views further extend the possibilities for model validation. Based on this integration, a discussion outlines steps towards supporting the entire workflow of identifying regression models. An evaluation illustrates a typical workflow in the application context of automotive engine design and reports general feedback of domain experts and users of the approach. These results indicate that the approach significantly accelerates the identification of regression models and increases the confidence in the overall engineering process.

An important aspect of HyperMoVal is to scale with respect to the dimensionality of the validated regression models. Even more importantly, however, HyperMoVal demonstrates how visual analysis is suitable to significantly improve a real-world task in a concrete application domain. Researchers have recently been stressing the importance of the characterization of real-world problems and the difficulties of a successful technology transition [187, 219]. Being distributed as part of a commercial software suite, HyperMoVal – as well as the contributions summarized in the previous sections 1.4.1, 1.4.3, and 1.4.4 – are examples of such a technology transition from science to industry and thus highlight the practical impact of this thesis.

1.5 Organization

The remainder of this thesis is organized as follows: chapter 2 surveys the state of the art in scalable visual analysis with respect to large multivariate data. The subsequent five chapters present the main contributions of this thesis. Concerning computational scalability, chapter 3 describes a generic architecture to facilitate the development of highly interactive visual analysis tools using multi-threading. Two variants of scatterplots address the visual scalability for continuous data (chapter 4), and for partly categorical data (chapter 5). While comparisons in these scatterplots are only qualitative, chapter 6 contributes an approach for quantifying subsets of the data by means of statistical moments for a potentially large number of dimensions. As another topic involving the scalability to a non-trivial number of dimensions, chapter 7 describes a design study for the validation of regression models as an important application of visual analysis. Chapter 8 describes conclusions and implications of this work. The thesis concludes with acknowledgements as well as an extensive bibliography.

Chapter 2

The State of the Art

Chapter 1 motivated scalability as a key issue of visual analyis in general, and described large data as a particular challenge with respect to visual and computational limits. This chapter surveys the state of the art in scalable visual analysis of data that is large either with respect to the number of data entries, or with respect to the number of dimensions.

In consistence with the contributions of this thesis, the focus of this survey is on multivariate data, including hierarchical, spatial, and temporal data. Scientific visualization of large data [101] (e.g., rendering of massive volumetric data or large vector fields) as well as information visualization of textual data or large networks are separate topics which are beyond the scope of this report. However, many of the discussed approaches are general and can also be employed in scientific visualization and information visualization of networks and textual data.

This survey structures the discussion by the employed approach, i.e., data removal, data aggregation, dimension reduction, coordination, data management, and parallelization. This structure facilitates to compare related approaches. The chapter concludes with a brief summary of approaches for other scalability issues than the sheer size of data.

2.1 Data Removal

This section summarizes approaches that reduce the size of a multivariate dataset by temporarily removing parts of it. The main distinction concerns the strategy which data to keep. Sampling refers to a random selection of a subset of the data whereas filtering deterministically selects a data subset that satisfies certain criteria.

While the approaches discussed in this section operate on data items (i.e., the rows of a dataset), removal may also apply to data dimensions (i.e., the columns). This is especially important for truly high-dimensional data. In some sense, even assigning a pair of dimensions to the axes of a scatterplot can be regarded as a kind of data removal of all other data dimensions. However, in order to make the discussion as coherent as possible, dimension reduction is surveyed as an own topic in section 2.3.

2.1.1 Sampling

Randomness has gained an increasing importance in computer science for problems which are intractable for a non-trivial amount of data using deterministic approaches – including NP-

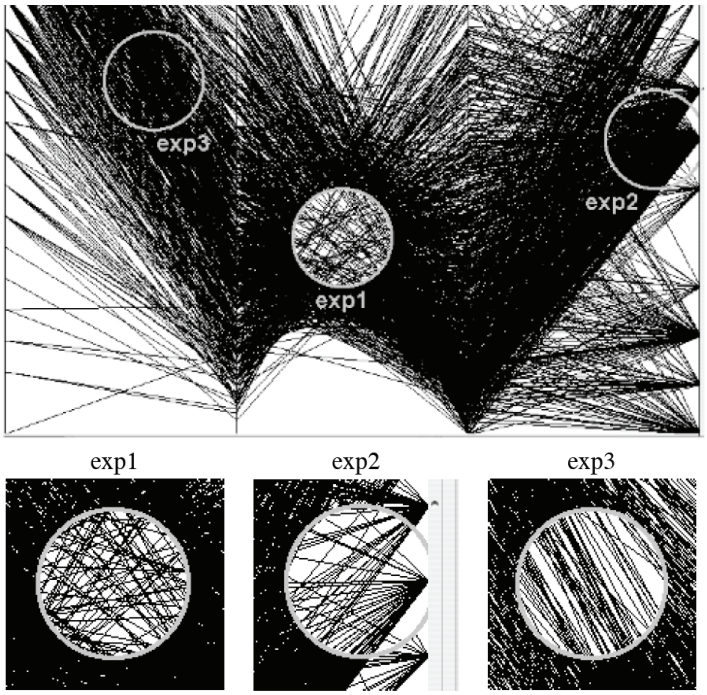


Figure 2.1: A lens-metaphor to locally reduce clutter by means of sampling [63].

hard ones. For example, genetic algorithms [92] enable to find a good solution in acceptable time rather than the optimal solution.

While computational limits also play a role in some visualization algorithms, the main motivation for using sampling in multivariate visualizations is to achieve visual scalability by the reduction of clutter [64]. Dix and Ellis [62] argue that random sampling can improve visualization algorithms (1) if calculations imply that information is lost anyway, (2) if there are too many data points to show, or (3) if details are only required for some data items. For visualizations based on aggregate or summary statistics (e.g., histograms), Dix and Ellis claim that sampled data can always be used to give approximations. For item-based visualizations (e.g., scatterplots or parallel coordinates), sampling the data will reduce overplotting in dense areas and thus make such visualizations more readable.

A key trade-off of sampling is to minimize the sample size while preserving as much accuracy of the visualization as possible. Dix and Ellis discuss respective issues involving perceptual limits, performance, and the interplay between common interactions like zooming and the visualization of sampled data [62]. They also stress the importance of correct sampling which means that sampling should preserve the statistical properties of the original data. This involves avoiding any bias towards particular values or categories.

Sampling can be applied globally or locally. Global sampling affects the entire visualization and the density is more or less independent of user interaction. For example, Bertini and Santucci propose a formal framework to measure the degree of visual overlapping, to obtain precise quality metrics about the visualization degradation, and to devise automatic sampling strategies in order to improve the overall image quality of 2D scatterplots [21]. In contrast, local approaches typically employ the lens-metaphor [22] to let the user reduce clutter in particular regions of the visualization. Ellis and Dix, for example, measure occlusion in

parallel coordinate plots to automatically adjust the sampling rate within the lens [63] (see Fig. 2.1).

Concluding, sampling has many advantages compared to other clutter reduction techniques [64]. In particular, it is scalable and preserves all information of kept data items. Reducing data at an early stage of the visualization pipeline, sampling may improve both information and visual scalability at the same time. However, sampling by itself neither guarantees that individual data items can visually be discriminated, nor does it necessarily convey the overlap density in overplotted views. Moreover, avoiding bias can be very difficult to accomplish for complex datasets that comprise multiple categorical as well as continuous data dimensions. Finally, whether sampling is a viable option at all depends on the task. While sampled representations generally preserve trends and correlations, they may discard important information if the user is looking for outliers or other single data items.

2.1.2 Filtering

Filtering is a technique to deterministically reduce the set of visualized data items based on some specific conditions. As a typical reason for filtering, parts of the data are sometimes considered irrelevant for a particular task, e.g., certain categories or values exceeding certain thresholds. Filtering these parts may significantly reduce both the computational and visual complexity of a visualization.

Historically, filter conditions were typically specified once at the very beginning of the visualization pipeline, i.e., at the time of data import as batch-oriented text queries (e.g., using the Structured Query Language (SQL)). In case of huge input data as for large data warehouses, such kind of static filtering is often still reasonable to allow for a subsequent interactive analysis of the data at a particular level of detail. However, more interesting in the context of visual analysis are interactive approaches to support filtering the data dynamically during the analysis. Several taxonomies describe filtering as a basic type of interaction, for example Shneiderman [227], Keim [148], and Wilkinson [277]. More recently, Yi et al. [286] defined filtering as "to show something conditionally".

Different types of data require different interaction techniques to specify filter criteria. Many systems (e.g., Spotfire [1]) reserve a dedicated area for dynamic query controls like range sliders for continuous or check boxes for categorical data attributes [2]. Prosection views may also employ range sliders to control the displayed parameter range for each parameter of a multi-dimensional function [258]. In contrast, the *Name Voyager* [271] supports filtering of names by entering the first letters through keyboard interaction.

Some systems integrate the specification of filter criteria in the visualization itself. For example, a common approach to filtering categorical data is to support dragging categories onto a dedicated panel (e.g., implemented by Polaris [238] and SellTrend [168]). Depending on the semantics of the panel, this either hides all data items of the respective category or everything except the selected category. For time-series data, Wattenberg proposed QuerySketch [269] which filters data items based on similarity metrics to a graph that users are able to draw freehand. Hurter et al. [122] proposed an interesting technique to filter thousands of aircraft trajectories. Users can spread the dataset across multiple views using pick and drop operations of selected trajectories, where each view only shows a subset of the data.

In most systems, filtering has a global effect as it affects the entire visualization of potentially multiple views (see section 2.4). In contrast, the Movable Filter [241] locally changes the view of objects within an arbitrarily-shaped region.

As a conclusion, filtering is appropriate if parts of the data are known to be irrelevant. As a major advantage of filtering, all information of the relevant data is preserved. The introduction of potential bias depends on the user-defined filter criteria rather than on a sampling strategy which can make efficient implementations much easier. Unlike sampling, filtering data may also be reasonable when looking for outliers. However, like sampling, filtering can not ensure the discrimination of individual data items in the visualization [64]. It is often not possible to obtain a certain target size if the relevant amount of data is still too large. Finally, if the user does not know precisely which data might be relevant, filtering (too much) data means losing potentially valuable context information. Therefore, highlighting interesting data by means of selection is often a reasonable alternative to filtering data (see section 2.4).

2.2 Data Aggregation

The term *aggregation* is defined as forming a whole by combining several separate elements [49]. In the context of data reduction, aggregation is a transformation in order to reduce the amount of detail while preserving certain important information. In contrast to data removal, typically all data items contribute in some way to the result of the aggregation.

This section summarizes aggregation-based visualization approaches. The discussion is structured by the type of aggregation, i.e., pivotization, binning, and abstraction. Finally, a separate sub-section is dedicated to aggregation of spatial, temporal, and spatiotemporal data in order to account for the importance and the special characteristics of such data. As for section 2.1, all approaches discussed in this section operate on data items (i.e., the rows of a dataset) while aggregation-based approaches operating on data dimensions are covered by section 2.3.

2.2.1 Pivotization and Hierarchical Structuring

Data dimensions of multivariate datasets can roughly be distinguished as being either continuous or categorical. In case of datasets having both categorical and continuous attributes, *pivot tables* have long been used to summarize the values of the continuous attributes with respect to a classification as given by the categories of categorical attributes (also referred to as conditioning categories). The concept of pivoting data is very important for databases, where the predominant Structured Query Language (SQL), for example, offers the “GROUP BY” clause of “SELECT” statements for this purpose.

Flat pivot tables can be visualized using common techniques for multivariate, quantitative data. The Gapminder Trendalyzer [89], for example, maps two aggregated indicators of countries to the axes of a time-dependent scatterplot and shows the population, i.e., the size of the category, by the area of according circles. However, categorical data is closely related to hierarchical data and pivot tables are often structured hierarchically. Apart from inherently hierarchical categories (e.g., years can be subdivided into months, days, hours, etc.), *dimension composition* defines hierarchies by specializing the categories of one attribute by the categories of another one. For example, two separate attributes ”sex” and ”age group” can be combined to obtain a category like ”female and younger than 30”. This is the key idea behind On-Line Analytical Processing (OLAP) [45] which typically uses large-scale overview summaries of the data as starting point for selective drill down into interesting parts of the data. OLAP is thus related to navigating a hierarchy. Gray et al. [94] proposed to

Figure 2.2: Tableau is based on a formal algebra to create visualizations intuitively using data pivotization [169].

treat multidimensional databases as n-dimensional data cubes to overcome the limitations of traditional SQL statements with respect to drill-down and roll-up operations.

While most OLAP front-ends only offer selected business graphics, Polaris [238] uses a formal algebra as specification of pivot tables and their visual representation. The user can incrementally construct complex queries by intuitive manipulations of this algebra. Stolte et al. [239] also describe an extension to the algebra for rich hierarchical structures. The layout of Polaris is based on small-multiple displays of information [256]. This refers to arranging the conditioning variables into rows and columns of a matrix where each cell visualizes a conditioned subset of the data. Polaris is a very intuitive and highly effective approach for analyzing data cubes, as shown by the success of its commercial version Tableau [169] (see Fig. 2.2).

As discussed above, incremental pivotization of data is equivalent to defining a hierarchy. There has been very much research on the visualization of hierarchies and hierarchically structured data (see Fig. 2.3). Containment-based approaches like tree maps [225] are one of the most popular techniques and show the size of the hierarchy nodes very well, while depth information is occasionally harder to read. In contrast, node-link representations [12, 116] show the structure more explicitly, but most approaches do not clearly convey the size of the nodes. The rooted tree growing from top to bottom is a very common layout, but does not utilize space efficiently for large hierarchies. Centric approaches are superior in this respect as they grow outwards from the representation of the root node and thus allocate more space to more detailed levels of the hierarchy. Nodes are typically placed in correspondence

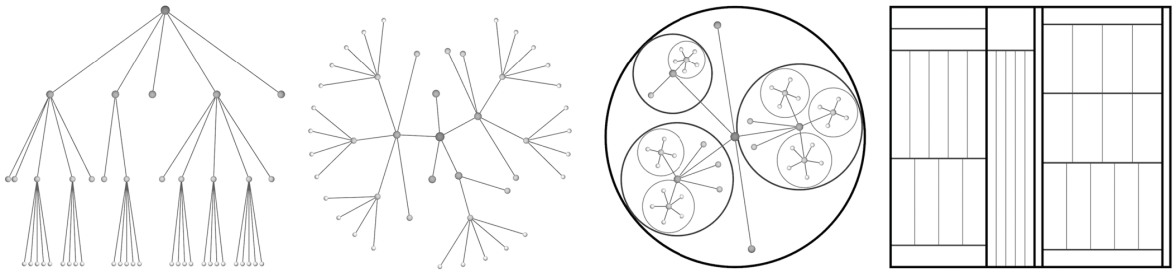


Figure 2.3: Different tree visualization techniques. From left-to-right: rooted tree, radial tree, balloon tree, and treemap layout [119].

to their position in the hierarchy, e.g., putting nodes with equal depth on concentric circles (radial tree) [12] or enclosing each sub-tree in a bubble (balloon tree) [116]. There are many extensions and variations to these approaches: focus+context techniques to improve scalability [161], combinations of node-link representations and enclosure [289], combinations of centric layout and enclosure [284], and edge bundles for integrating relations between items into the visualization [119].

Recently, Slingsby et al. [231] explored the effects of selecting alternative layouts in hierarchical displays that show multiple aspects of large multivariate datasets. They employ size, shape, and color to show subset properties and order the position of the hierarchy nodes by the conditioning variable values using dimensional stacking. Slingsby et al. point out that the use of different layouts at different hierarchical levels can help to use the coordinates of the plane more effectively in order to draw attention to trends and anomalies in the data.

Concluding, pivotization may significantly reduce the amount of discriminated entities for subsequent steps like visualization. In contrast to sampling and filtering, all data entries contribute to the final result even in case of billions of data records. Dimension composition supports a controlled and semantically meaningful adjustment of the degree of detail using drill-down and roll-up operations. This might be the reason why OLAP has become the standard approach to business intelligence (and other applications) in the past decade. However, these benefits come at the cost of a potentially huge loss of detail when condensing the values of multiple entries to a few summary statistics – typically simple univariate moments like the average or the maximum. Moreover, pivotization changes the unit of the entities in terms of which the data is being analyzed. For example, after pivoting a financial dataset logging individual transactions by the respective stock, any visualization will compare stocks rather than transactions.

2.2.2 Binning

Item-based visualization techniques for multivariate data (e.g., scatterplots and parallel coordinates) are in general more suitable for continuous data attributes which make more efficient use of the available space. On the other hand, it can be advantageous to categorize continuous data attributes before the visualization in case of a very large number of data items. This process is referred to as binning which converts continuous data to a frequency-based representation by dividing the data space into a set of intervals - called bins - and assigns to every bin an occupancy value which determines the number of data records that belong to the bin [229]. In the context of visualization, the primary goal of binning is to reduce visual

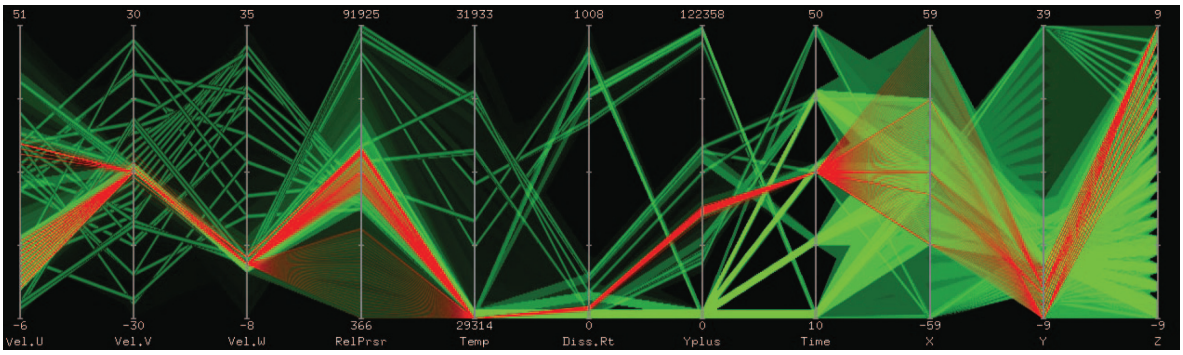


Figure 2.4: Outlier-preserving focus+context visualization of a flow simulation dataset [194].

clutter and to preserve the distribution characteristics of the data which may otherwise be concealed due to overplotting.

Histograms, for example, typically employ univariate binning with equally sized bins and they represent the number of data items in each bin by the height of a respective bar. Other visualization techniques use multidimensional binning. Some extensions of histograms generate a 2D map of two binned variables (e.g., time histograms [153]). This map can be displayed either in 3D as a height field or in 2D as an image with color representing the density. Novotny and Hauser generate parallel coordinates on the basis of a binned data representation for what they call an output-oriented visualization approach [194] (see Fig. 2.4). Their approach draws parallelograms instead of single lines for the two-dimensional bins of each pair of adjacent axes. Furthermore, they also discriminate between outliers and trends based on clustering as performed on a binned data representation (see section 2.2.3).

Other approaches adapt the size of the bins to the characteristics of the data. Hao et al. [103] recently proposed variable binned scatterplots to allow the visualization of large amounts of data without overlapping. The basic idea is to use a non-uniform (variable) binning of the x and y dimensions and to plot all data points that are located within each bin into the corresponding squares.

Generally speaking, the visual scalability of a binned visualization typically depends on the number of bins rather than on the size of the data which is an interesting aspect for large datasets. Varying the number of bins is the key to controlling the amount of details. Bin sizes that correspond to a single pixel in image space hardly incur a visible loss of detail and may reduce the effort for other involved computations. On the other hand, binning may introduce aliasing artifacts and interaction techniques like zooming may require a frequent re-binning of the data. Finally, binning is not easily applicable to all types of visualizations (e.g., glyph-based visualizations).

2.2.3 Abstraction

The objective of data abstraction is to convey the important information while suppressing irrelevant details [247]. The key idea is to compute values or patterns which can then be used for further analysis or visualization processes rather than the raw data.

Descriptive statistical moments are a common type of abstraction in the context of data aggregation. Box plots have a long history and they are still one of the most common approaches to graphing summary statistics [257]. The standard box plot summarizes the distribution of

a dataset by its minimum and maximum range values, the upper and lower quartiles, and the median. Many extensions have been proposed to include additional information like the density of the data. Potter et al. [205] provide a summary of these extensions and propose a new hybrid summary plot that includes additional statistical quantities like the skew or the tail in order to convey certain aspects of data uncertainty. Kehrer et al. [142] also include higher-order statistical moments like the skewness and the kurtosis as well as robust estimates in an iterative visual analysis process. The focus of their work is on integrating statistical aggregations in a framework of coordinated multiple views (see section 2.4) by enabling an analyst to brush particular statistics.

Clusters are another popular type of abstraction for multivariate data. A cluster refers to a subset of data items that are similar in some sense. The process of cluster analysis partitions a collection of data items into separate groups with respect to a given measure of similarity [254]. Cluster analysis belongs to the field of unsupervised learning because it does not require pre-classified training data. Hastie et al. [106] provide a comprehensive overview of statistical learning in general and unsupervised learning in particular. However, Nam et al. [188] point out that results from fully automated cluster analysis often do not match the knowledge and intuition of domain experts. Therefore, Nam et al. describe a framework that integrates the user in the derivation of classification hierarchies. This framework enables users to interactively tune parameters of k-means clustering based on a visualization of the inherent characteristics of the data in high-dimensional space.

Numerous visualization approaches have been proposed in the recent years that build on the results of clustered data. A common goal is to tackle problems caused by clutter when drawing a large number of data items. For example, Fua et al. [82] propose a hierarchical version of parallel coordinates which is based on a multi-resolution view of the data via hierarchical clustering. Yang et al. [285] generalize this idea to a general framework for interactive hierarchical displays. In order to convey aggregation information about the resulting clusters, the authors describe hierarchical variations of traditional multivariate visualization techniques including star glyphs and scatterplot matrices.

Especially parallel coordinates have seen multiple variations to improve the representation of higher dimensional clusters. Most of these variations have recently been surveyed and evaluated by Holten and van Wijk [120]. As two examples of variations, Johansson et al.[134] use transfer functions operating on high-precision textures to highlight different aspects of the cluster characteristics. The authors also apply feature animation as a guidance when simultaneously analyzing several clusters. Zhou et al. [290] exploit curved edges to form visual bundles for clusters in parallel coordinates in order to increase their chance of being revealed.

Besides cluster analysis, numerous other types of abstraction models may be used to represent different kinds of trends in the data. *Regression models* are perhaps among the most important types of statistical models, establishing a linear or non-linear relationship between one or more independent parameters and a continuous result dimension [59]. Lines and curves have long been used in statistical visualization to display regression models in 2D scatterplots. Going beyond global trends, Reddy et al. [208] describe *data skeletons* as a visual abstraction for high-dimensional data distributions that may contain local trends in certain subspaces. The idea is to represent certain shapes of data by a graph that consists of segments of locally fit principal curves or surfaces summarizing each identified branch.

In contrast to trends, *outliers* have not yet attracted as much attention in visualization research. This is somewhat surprising, as many applications (e.g., intrusion detection) may

regard outliers as the most interesting part of the data. As one exception, Novotny and Hauser [194] explicitly separate outliers from trends in order to treat them differently for visualization in parallel coordinates according to their different characteristics and semantics (see Fig. 2.4). Kehrer et al. [142] also include measures of outlyingness in their study of opportunities for the interactive visual analysis of multi-dimensional scientific data.

Concluding, abstracting from raw data is a key approach to generate effective visualizations even for huge datasets. In some cases, well-designed abstractions have the potential to avoid clutter and to convey important aspects much better than it would ever be possible by showing even parts of the raw data, which makes abstraction a key issue for visual scalability. These benefits may have also been an important reason why Keim reformulated Shneiderman's well-known information seeking mantra ("overview first – zoom/filter – details on demand" [227]) to "analyze first – show the important – zoom, filter and analyze further – details on demand" [147].

On the other hand, abstraction also implies a loss of detail that is not always tolerable. Some abstraction methods may require significant computational resources which may contradict the requirement of highly interactive tools. Finally, especially complex models like clusters and non-linear regression models may not reasonably be identified in a fully automated way. Building such models typically requires multiple iterations including reparameterization and validation based on the knowledge of domain experts for the data. As discussed in chapter 1, this fact is in accordance with the goal of visual analytics to strive for a tight integration of interactive visualization and automated data analysis [247]. However, it may be a prohibitive disadvantage if the task of the user is different than model building as such.

2.2.4 Aggregation of Spatial and Temporal Data

Temporal references and spatial coordinates are often treated just like ordinary numeric variables. However, temporal and spatial data have several specific characteristics that distinguish them from other types of data [8]. For example, characteristics at proximal locations tend to be correlated [249] which is often called "the first law of geography". Concerning time-oriented data, Aigner et al. discriminate between a linear versus a cyclic view on time, between point-oriented versus interval-oriented temporal entities, and between ordered time versus branching time [3]. Therefore, this sub-section briefly summarizes aggregation concepts that are specific to temporal, spatial, or spatiotemporal data.

A common way to represent time is using a hierarchy of *temporal categories* (e.g., years, months, days, hours, etc). Such a representation is a specific type of data cube and is thus frequently used for pivotization (see section 2.2.1). Tableau [169], for example, intuitively supports navigating temporal hierarchy levels to adapt the degree of detail (see Fig. 2.2). Tableau also demonstrates how a flexible assignment of temporal categories to visual attributes enables the user to switch between a linear and a cyclic view on time. Assigning all temporal categories to a single axis (typically the X-axis) generates a linear view on time; assigning temporal categories to multiple axes supports the comparison of cycles (e.g., years being represented as rows and months being sequential columns). Wang et al. propose temporal summaries [266], an interactive visualization technique that dynamically aggregate events in multiple granularities for the purpose of spotting trends over time and comparing several groups of records.

Apart from common descriptive statistics like maximum or average, an interesting option

to aggregate a part of a time-series to a single quantitative value is by its *similarity* to a certain temporal pattern. This is the key idea of the TimeSearcher [118]. Muigg et al. [186] also employ such an aggregation for brushing a subset of many function graphs. Van Wijk et al. [262] use similarity for clustering daily patterns in order to identify patterns and trends on multiple time scales simultaneously.

The *temporal context* is often important for data abstraction. Aigner et al. [3] distinguish between basic and complex temporal abstraction methods. An example of a basic abstraction method is a qualitative classification of the gradient of a time-dependent value (e.g., "decreasing fast"). VIE-VENT [183] is a system that provides more complex temporal abstraction like context-sensitive and expectation-guided methods in a medical application domain. The methods incorporate knowledge about data points, data intervals, and expected qualitative trend patterns to arrive at unified qualitative descriptions. As another example, The Spread [184] implements a time-oriented data abstraction method to derive steady qualitative descriptions from oscillating high-frequency data.

Much work has been dedicated to the visualization of *geographical data*. Dykes et al. [60] give a survey of the existing systems and future trends of geo-visualization. With regards to aggregation, hierarchical categories also play an important role for geographical data, (e.g., continents, countries, states, counties, etc.). When aggregating geographical data, it is essential to consider that the analysis results may depend on how the units are aggregated. This refers to the sizes of the aggregates (scale effects) as well as to their locations and composition from smaller units. As an example, Shanbhag et al. [224] use visualization of demographic data over time to validate partitions. Chang et al. [38] present an interactive tool for urban visualization that aggregates buildings and city blocks into legible clusters. The goal is to provide continuous levels of abstraction while preserving the users mental model of the city.

In many cases, data has both spatial and temporal characteristics. Such *spatiotemporal data* can be regarded as spatial distributions (situations) changing over time or as profiles of local temporal variation distributed over space. To support both analytic perspectives, Andrienko et al. [5] recently suggested a framework based on Self-Organizing Maps (SOMs) combined with interactive visual tools. The authors apply SOMs to spatial situations at different time moments as well as to local temporal evolution profiles. The SOM matrix displays groupings of data objects and their two-dimensional arrangement by similarity.

Movements (e.g., trajectories of cars) are a particularly important type of spatiotemporal data. Large sets of movement data typically involve data aggregation that can be spatial (S), temporal (T), attributive (A), or a combination thereof [78]. As an example, an SSTT aggregation (start place, end place, start time, and end time) counts for each pair of places in space the number of movements between two time moments. This can be visualized as a transition matrix [97]. Andrienko et al. [6] extend SSTT-aggregation by a route-based grouping of movements. In a later paper, Andrienko et al. [4] investigate the use of aggregation for two possible views of movement, traffic-oriented and trajectory-oriented (see Fig. 2.5).

As a conclusion, the variety of different aggregation methods for spatial, temporal, and spatiotemporal patterns shows the importance of considering the special characteristics of such data in the context of the task. The fact that most data have some type of temporal or spatial reference – also in information visualization – makes respective aggregation methods a major issue of scalable visual analysis.

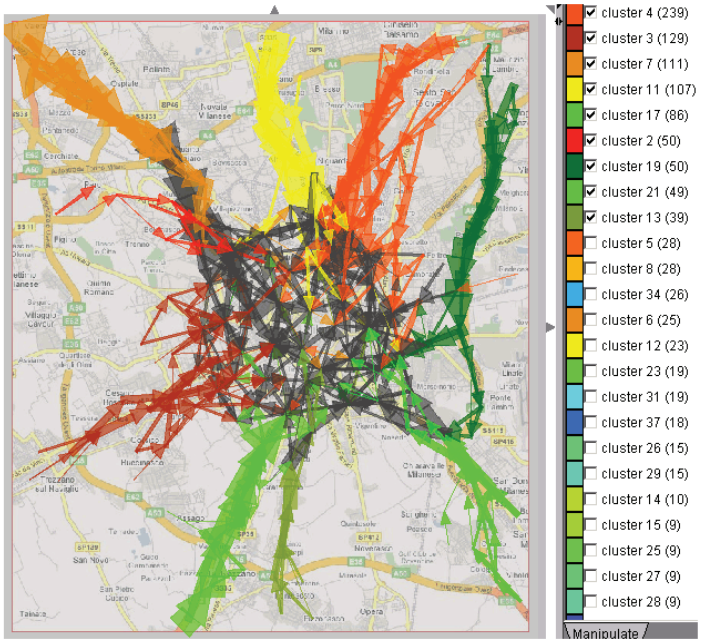


Figure 2.5: Different clusters of trajectories going to the center of Milan [6].

2.3 Dimension Reduction

All approaches discussed so far address scalability with respect to many data items, i.e., data tables with a large number of rows. However, also high-dimensional datasets are becoming increasingly common, i.e., data tables with a large number of columns. Examples include frequency spectrums in technical data and surveys with many questions.

Most visualization techniques for multivariate data have inherent or practical limitations with respect to the number of dimensions that can simultaneously be displayed. A basic 2D scatterplot, for example, scales up to a few thousand data items, but it displays only two dimensions. Parallel coordinates and scatterplot matrices become increasingly hard to interpret and scale up to 10 to 20 dimensions at best. Therefore, exploring truly high-dimensional datasets is a non-trivial yet important research question.

As one solution, there are numerous methods that project high-dimensional data to low-dimensional space. *Principal Component Analysis* (PCA) of N-dimensional data is an approach that generates a sequence of N best linear approximations to the data which are ranked by the amount of variance [138]. Each principal component is a linear combination of the original dimensions that is orthogonal to all other principal components. While PCA itself is a lossless transformation that does not reduce the number of dimensions, its application in the context of dimension reduction is based on the ranking of the principal components. In most cases, the first few components account for the majority of information in the data which can enable the detection of high-dimensional clusters in common scatterplots, for example.

Many *non-linear methods for dimension reduction* attempt to preserve similarities between objects in high-dimensional space after the projection to low-dimensional space. For example, Multi-Dimensional Scaling (MDS) [179] is an iterative approach to minimize a stress-function that expresses how much the distances between objects change by a projection to low-dimensional space. Self-Organizing Maps (SOMs) [151] perform vector quantization to

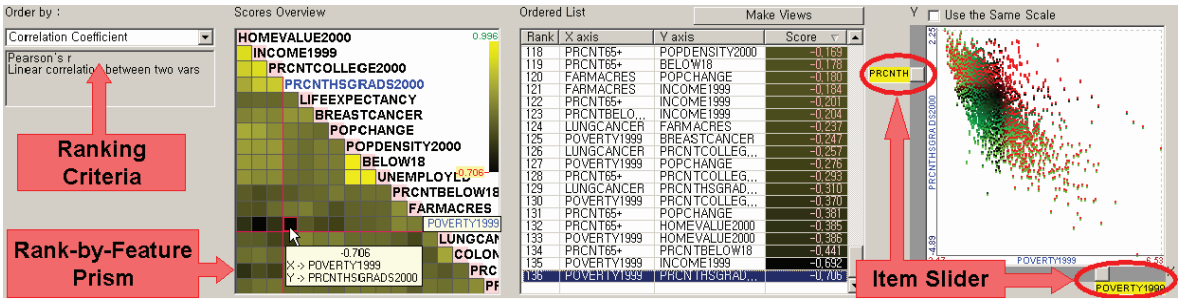


Figure 2.6: The Rank-by-Feature Framework for ordering pairs of dimensions by bivariate statistical moments [222].

project high-dimensional data vectors to a limited number of prototype vectors which are often arranged as a 2D regular grid. Visualization results often resemble landscapes that show clusters in the data as mountains [75].

Schreck et al. emphasize that one problem of MDS and SOM is the uncertainty regarding the precision of the projection as compared to the original data characteristics [220]. They address this issue by integrating a projection precision measure into the projection visualization and discuss several visual mappings for integration at various levels of abstraction. However, the main drawback of PCA, MDS, SOM, and all derivates (e.g., non-linear PCA and local MDS [106]) from a visualization point of view is the lack of intuitive meaning of the generated dimensions which makes results difficult to interpret. As one approach to mitigate this in case of PCA, Jeong et al. [132] describe a system that visualizes the results of PCA and which provides user interactions to assist in better understanding PCA.

As an alternative solution, several *interactive approaches to dimension reduction* have been published. Most of them focus on different structures within the data. Yang et al. propose visual hierarchical dimension reduction [283], which groups dimensions into a hierarchy in order to let the user select a meaningful set of dimensions from different clusters of the hierarchy. Later, Yang et al. extended their work to what they call interactive hierarchical dimension ordering, spacing and filtering approach (DOSFA) [282]. While DOSFA is also based on dimension hierarchies, dimension filtering automatically picks out important dimensions to form lower dimensional subspaces that contain major features of the original datasets.

Some approaches rely on *ordering dimensions* with respect to certain metrics. Peng et al. [201] define different measures of what constitutes clutter in terms of display properties for four different visualization techniques and they order the dimensions so that clutter is minimized. The Rank-by-Feature Framework [222, 223] ranks dimensions and pairs of dimensions by univariate and bivariate statistical moments (see Fig. 2.6). The user may choose between several statistics which are displayed in a linked table for ranking preview visualizations as guidance to potentially interesting dimensions. Johansson et al. [136] proposed to combine user-defined quality metrics using weight functions. Their approach supplies a range of automatic variable orderings and enables a quality-guided reduction of variables (see Fig. 1.2 (d)).

A different idea is to visually represent the space of dimensions itself. Ankerst et al. [7] introduced a measure to place dimensions with alike behavior close to each other. Based on a similar idea, Yang et al. [281] represent dimensions as pixel-oriented glyphs and apply Multi-Dimensional Scaling to position them in 2D space according to their properties (see

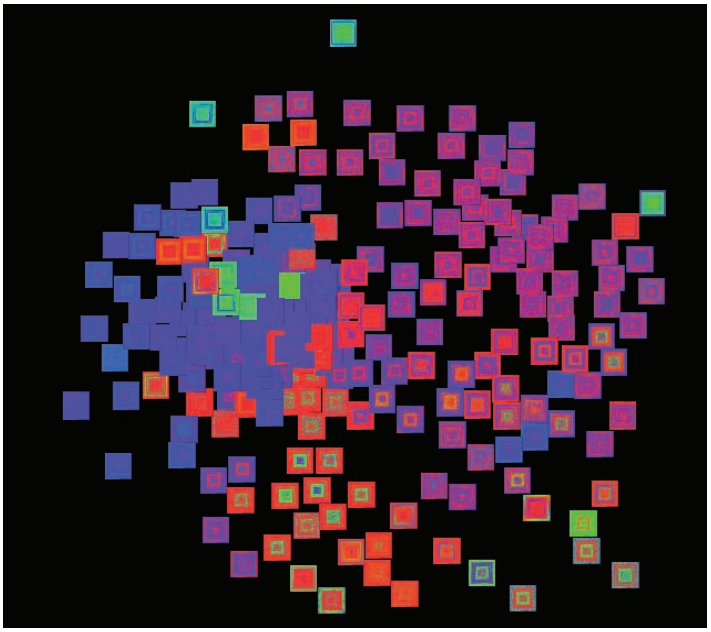


Figure 2.7: Representing each dimension as pixel-oriented glyph [281]. Multi-Dimensional Scaling [179] is employed to position the glyphs according to the correlation between the dimensions.

Fig. 2.7).

Another group of work assesses projections by different *user-dependent notions of interestingness*. Wilkinson et al. [278] apply Graph Scagnostics to characterize axis-parallel 2D projections by measuring properties like convexity, correlation, or outliers. Dasgupta et al. recently proposed Pargnostics [48] as a similar approach for parallel coordinates. Sips et al. [230] described two quantitative measures of class consistency to assess the quality of the mapping. Tatu et al. [244] measure interestingness of labeled and unlabeled point clouds to provide the user with a small number of potentially useful candidate visualizations.

As a conclusion, despite significant advances achieved by a wealth of both automated and interactive approaches, analyzing truly high-dimensional data will perhaps always remain a challenge due to the inherently limited intuition of humans for high-dimensional space. However, many approaches have successfully demonstrated that it is possible to partially solve the problem for specific tasks or data characteristics. An increasingly tight integration between interactive visualization and automated data analysis is likely to further increase the number of solutions to real-world problems that scale to high dimensionality.

2.4 Coordination

The analysis of complex data typically requires a variety of different components. Examples include different visualization types, data projections, interaction techniques, and automated methods for data analysis. Coordinating multiple components is thus a key issue in the context of scalable visual data analysis. Olson et al. describe coordination as composing purposeful actions into larger purposeful wholes, where the additional information processing performed when multiple, and connected actors pursue goals that a single actor [or indeed the multiple

actors working separately] pursuing the same goals would not perform [196]. Boukhelifa et al. [28] emphasize the point that the whole is greater than the sum of its components and they argue that any operation can be coordinated.

This section summarizes the state of the art in coordination for information visualization. Section 2.4.1 is dedicated to the coordination of multiple views, which is a very frequent type of coordination in systems for visual data analysis. Section 2.4.2 then surveys approaches that combine overview and detail information, which can also be considered a coordination of multiple degrees of visual detail.

2.4.1 Multiple Coordinated Views

The term multiple views is generally used to describe visualization approaches that employ multiple windows to show different representations of data [211]. Baldonado et al. [11] defined principles when and how using multiple views should be considered. They suggest to use multiple views in case of diversity in attributes, abstractions or genres when these views draw out correlations or disparities. Moreover, they point out that smaller manageable views can be helpful to partition complex visualizations.

The need to coordinate many components in a flexible way incurs a significant additional complexity to the design of a system for effective visual data analysis. Based on examples from visualization literature, Roberts identified multiple basic architectures to achieve view coordination [211], namely constraint-based [177], data-centric [192], and based on the Model View Controller (MVC) paradigm [200]. Boukhelifa et al. [28] argue that models for coordination allow for an effective development as well as a qualitative evaluation of systems that incorporate coordination. To this extent, they describe an abstract model that is based on sharing objects such as the visualization parameters of the dataflow model to achieve coordinated exploratory tasks in multiple views. Weaver proposed a similar conceptual model called Live Properties [272], in which views are tightly coupled through shared objects that describe the data contents and the navigation and selection state of views in a visualization.

There are many types of multi-view-environments and multiple different characteristics to discriminate them. One decision concerns the scope of updates when parameters change. In his model, Roberts distinguishes between replacement, replication, and overlay [210] with replication being the principal model of coordinated multiple views.

A key distinction of multi-view-environments is the strategy for *managing views*. Dual view systems [47] combine only two views on the data. A typical example is providing separate views for overview+detail, and also focus+context can to some degree be regarded as a special type of dual view (see section 2.4.2). Small multiples [256] are another type of multi-view-environment. Their main purpose is to enable visual comparisons by repeating a series of related views. Scatterplot matrices [43], for example, lay out all two-dimensional projections of a high-dimensional dataset as a grid so that adjacent plots share a common axis. HyperSlice [261] uses the same layout to visualize high-dimensional scalar functions. Other examples of small multiples subdivide the space according to categorical dimensions. Polaris [238] employs a regular subdivision while hierarchical layouts like tree maps [225] scale cells to represent a certain property for each node. In some sense, also pixel-based visualizations [143] can be considered as an extreme case of small multiples in which "views" have been reduced to single colored pixels that convey information when being arranged with respect to a certain criterion.

For more general multi-view-environments, various strategies exist to simultaneously man-

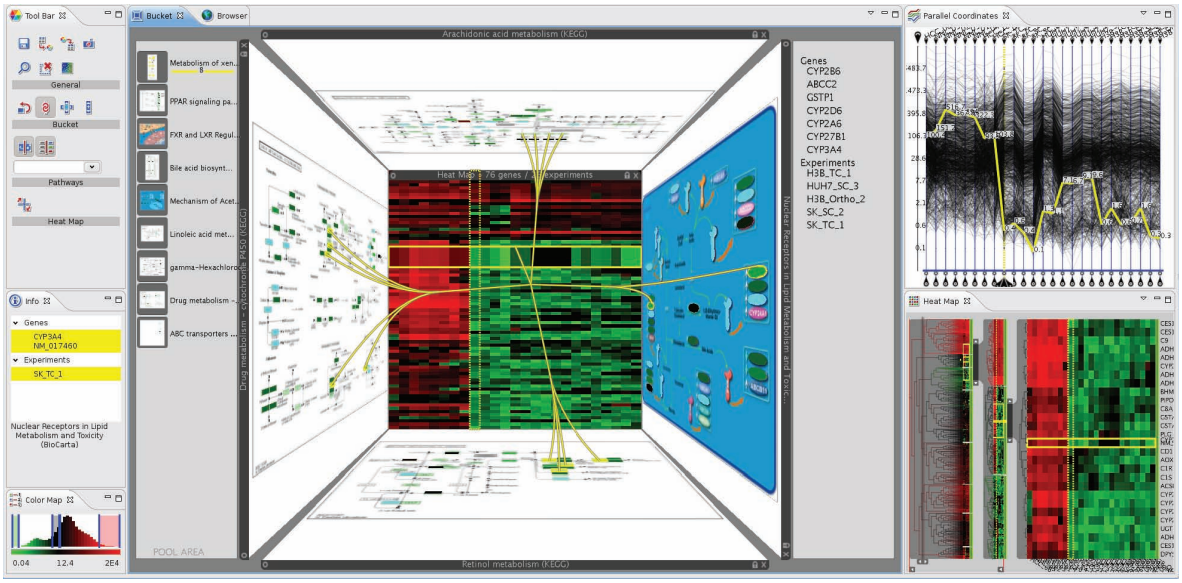


Figure 2.8: Caleydo [166] as an example of multiple coordinated views. The bucket layouts multiple views in a 2.5D setting. VisLinks [46] connect related information across views.

age diverse types of views. Simple ones may rely on the operating system to layout different windows or may utilize well-known dynamic splitters. More complex approaches display iconic representations to represent a navigable history of an analysis session [159, 113]. Lex et al. [166] layout multiple views as a bucket, i.e., the inside faces of a cube (see Fig. 2.8). This approach puts the focus on the center view while showing the surrounding views as context information. The 2.5D setting enables to intuitively draw connections (VisLinks [46]) between related visual representations in different views.

Another characteristic aspect that is tightly related to the strategy for managing multiple views is the interaction concept for *creating new views*. As the most simple case, many systems offer only a fixed layout and do not allow the user to create new views at all [168, 55]. Many other systems rely on standard menus and buttons for this type of interaction (e.g., Spotfire [1]). As an example of a more advanced approach, FromDaDy [122] enables the user to create new views by splitting the data shown in one view through interactive selection. Tableau utilizes an algebra in combination with a predetermined ranking of different visualization types in order to automatically create and parameterize small multiple views [169] (see Fig. 2.2). Other approaches offer module visualization environments that enable users to individually combine different visualization components also to define precisely how these components should be linked to each other (e.g., Snap-Together [192] and Improvise [272] (see Fig. 2.9)).

Besides the aspect of creating views, interaction in a general sense plays a key role for coordinated views [286]. Roberts distinguishes between direct and indirect interaction techniques for coordination [211]. *Indirect interaction* refers to approaches where the user interacts outside the visualization, e.g., dynamic queries for filtering data items (see section 2.1.2). Although not strictly interactive, also simultaneous animation in multiple views with respect to a common temporal reference can be regarded an indirect coordination technique [213].

In contrast, *direct interaction* techniques are performed within the visualization. Examples

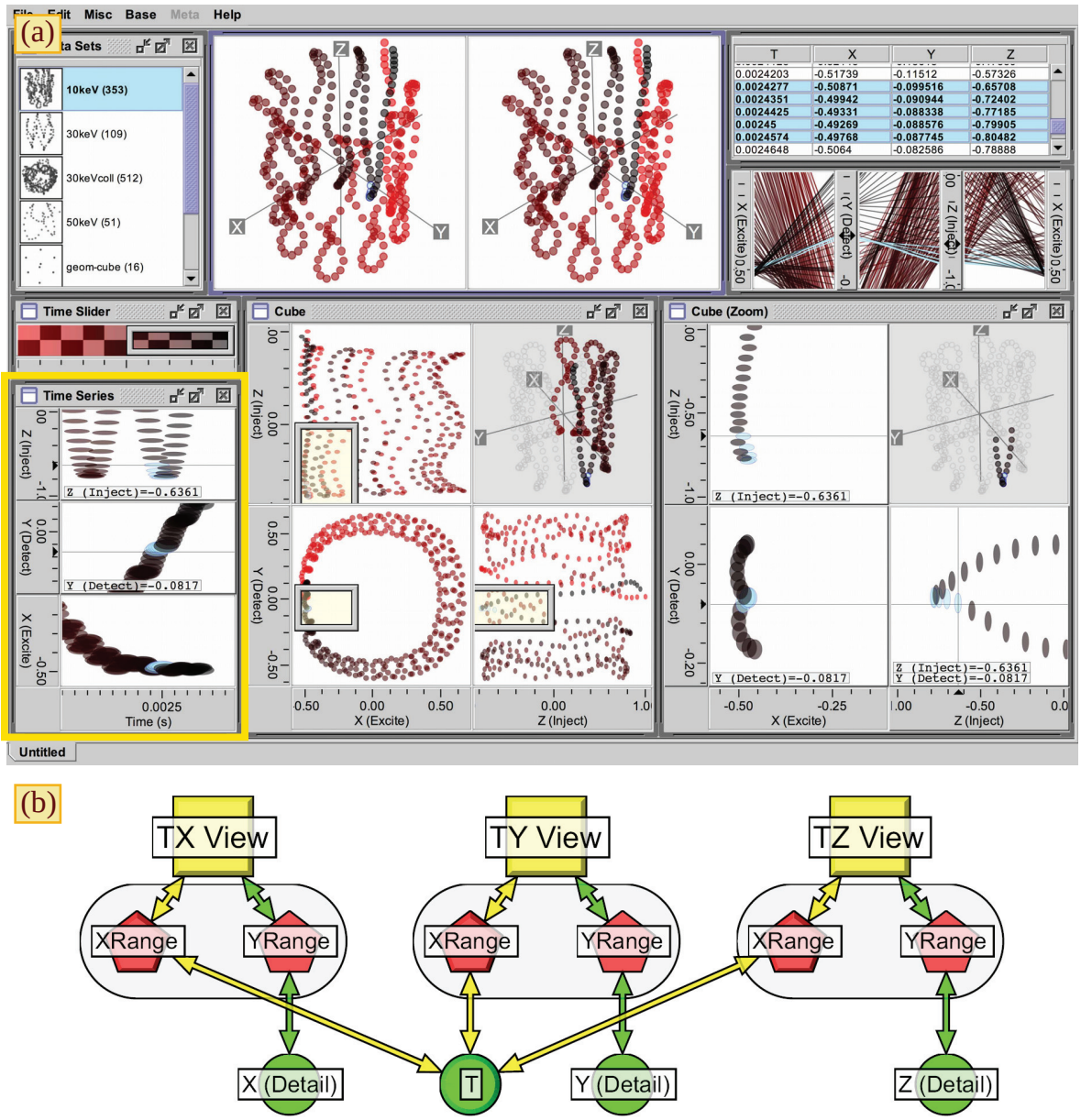


Figure 2.9: (a) Example of Improvise [272]. (b) The coordination pattern of the three highlighted scatterplots in (a) synchronizes their horizontal scrolling.

of such techniques include linked cursors [72], linked navigation [203], and data probing [32]. The most common type of direct interaction technique, however, is brushing which refers to defining different degrees of interest in the data. Brushing techniques are mostly specific to a certain type of visualization or data and differ in their complexity. Simple approaches include selecting individual data items by clicking on them, or marking coherent axis-aligned rectangular regions in scatterplots [16]. Examples of more advanced approaches involve angular brushes in parallel coordinates [108] and brushes of temporal characteristics [118, 186].

Complex tasks often require a more expressive way of formulating the degree of interest than it is possible by an isolated application of individual selection techniques. In this case, logical operations can be used to compose queries of arbitrary complexity [274]. Doleisch et al. [52] support smooth transitions in the degree of interest and apply fuzzy logic [288] for composite queries (see also chapter 4). In accordance with the idea of visual analytics [247, 146], a recent trend is to increase the intelligence of the data selection process. Hao et al. [102], for example, identify relationships to portions of the data which are similar to what the user has interactively selected before. Similarly, Yang et al. [280] presented an analysis-guided exploration system that extracts pieces of valuable information (called nuggets) based on the interests of users and also helps to organize a growing pool of such nuggets.

As a conclusion, coordinating multiple views is a suitable – and often necessary – approach to address complexity in data analysis. As described by Baldonado et al. [11], this complexity often results from diversity in the data that would be inappropriate or impossible to represent in a single view. Therefore, coordinated multiple views are a key technique with respect to visual scalability. There are several studies evaluating perceptional aspects in multi-view environments which illustrate the benefit for certain tasks, e.g., as compared to zooming [204] (see section 2.4.2).

However, Baldonado et al. also warn that multiple views should be used sparingly and that sequential comparisons can sometimes be favorable to side-by-side visualizations [11]. Furthermore, they stress the importance of consistent interfaces across views and the necessity of perceptual cues to make relationships between views more evident. Recently, Liu and Stasko [167] noted that internalization is a necessary first step in order to build a mental model when learning a visualization system. Many users still do not have such a mental model of the concept of multiple coordinated views (see section 7.5). Therefore, using – and even more so defining – multiple views is a mentally complex process for novices that may take a considerble amount of time to familiarize with.

2.4.2 Overview and Detail

As motivated in chapter 1, it is typically not possible to visualize all details of a large dataset in one image. As a solution, many approaches let the user specify a different degree of interest in different parts of the data [107]. Such a definition can be used to provide details only for interesting parts of the data. However, in contrast to filtering techniques that typically eliminate the rest of the data (see section 2.1.2), this section surveys approaches which combine the visualization of detail information with a representation of an overview of contextual data or even the entire dataset. The purpose of the overview is to provide contextual information and to let the user keep track of the current position in the data.

Several taxonomies and books identify different classes of approaches for providing overview and detail (e.g., [155, 236, 44]). Most of this work discriminates the subsequent categories which are also used to structure the discussion in this section:

- *Overview+detail.* Such interfaces show the overview information separately from detail.
- *Zooming.* An interaction technique to achieve a temporal separation between overview and detail.
- *Distortion-oriented focus+context.* Such techniques distort the image geometrically.
- *In-place focus+context.* Respective visualizations use visual cues to emphasize certain parts of the data.

Overview+detail interfaces spatially separate the display of the overview from the detail information. Examples dating back to the early 1980s are computer games (e.g., Defender), which display a low-resolution map of an entire level as overview in which a rectangle indicates the area that is currently shown in full-resolution. Very similar ideas can nowadays be found in many commercial applications, e.g., image editing software like Adobe Photoshop or geographical software like Google Maps. In information visualization, an early example is the Information Mural [133], where a smaller overview window shows the structure of the entire dataset while the detail area is given more screen space. A key challenge for using overview+detail in information visualization as compared to applications for rasterized images is to generate a meaningful abstraction as overview. Chen et al., for example, recently proposed an approach to dynamically adjust the level of abstraction for overview dendrogram visualizations [40].

Lam et al. evaluated the effectiveness of overviews in interfaces that provide multiple visual information resolutions for a given set of tasks [160]. They found that participants of their study used overviews only for simple visual targets with a small visual span. As their conclusion, interactions across multiple resolution interfaces incur a high cognitive load which is a considerable barrier to an effective use. However, other literature comparing interfaces with and without overviews has reported benefits of overviews for faster navigation [14], to maintain orientation [202], and to make decisions about future navigation [121].

As an alternative approach, *zooming* achieves a temporal separation between overview and detail visualizations. Users can adjust the level of detail by magnifying or de-magnifying a visualization in place rather than seeing multiple resolutions simultaneously. Furnas and Bederson proposed Space-Scale diagrams [87] as a framework to address domain independent design challenges when conceptualizing navigation using zoom and pan. Van Wijk and Nuij presented a generic model to achieve both smooth and efficient animations when navigating large 2D information spaces [263]. Their model uses a metric describing the perceived velocity for simultaneous zooming and panning to derive solutions for animations between two views, automatic zooming, and the parameterization of camera paths. Concerning the effectiveness of zooming, Plumlee and Ware presented a model predicting the user performance for tasks requiring frequent movements and scale changes for zooming as compared to using multiple views [204]. Since their evaluation showed much higher error rates for zooming, they proposed the design heuristic that extra windows are needed when visual comparisons involve patterns which are too complex to be held in visual working memory.

A very important class of techniques – generally referred to as *distortion-oriented focus+context* – transforms the image geometrically to integrate both overview and detail within a single visualization. In the early 1970s, Farrand was amongst the first to present the use of different magnification factors [68]. About a decade later, Furnas' Fisheye view [84] is now considered as the start of computer-based focus+context visualization. The idea has soon

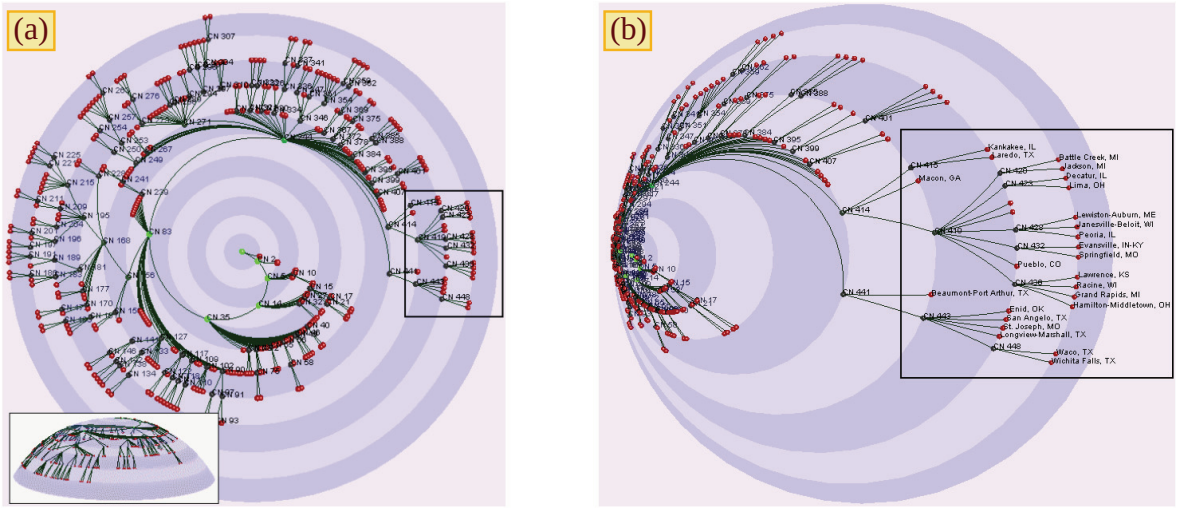


Figure 2.10: Projections onto a sphere enable a smooth shift of the focus [158].

been generalized [85]. An early taxonomy by Leung and Apperley [165] distinguishes three classes, namely continuous magnification, piecewise constant magnification, and others.

As an example for continuous magnification, Lamping et al. [161] draw in hyperbolic space and project the result to Euclidean space. Similarly, a projection onto a sphere enables the user to move the focus to all sides of the display [158] and supports a smooth degree of distortion (see Fig. 2.10). Keahey and Robertson describe non-linear magnification fields as a generalized technique to control the magnification function on a grid in a flexible way which also supports multiple foci [141]. While it is generally difficult to preserve shapes for large differences in magnification factors, Boettger at al. more recently proposed a technique to keep shapes intact and recognizable [27]. They use the complex logarithm and the complex root functions in order to show very small details even in very large contexts.

The concept of focus+context techniques with continuous magnification has been successfully adopted by widely-used commercial software especially for widgets offering a choice from a linear set of options, e.g., fish-eye menus [17] like the dock icon panel of Mac OS X. However, a continuously changing degree of distortion hampers the perception of distances for many other tasks. This is a major drawback for many types of visualizations like maps or scatterplots. In this respect, a piecewise constant magnification as employed by the bifocal display [237] preserves distances partially at least. A similar approach is the Perspective Wall [170] that does not distort the focus and bends the context backwards to both sides. Signal Lens [149] are an example of a very recent application of this concept to electronic time series.

The idea of distortion-oriented focus+context can both be specialized and generalized in many ways. Specializations mostly refer to extensions which are tailored to a specific kind of visualization. For example, various focus+context approaches have been proposed for visualizations of hierarchical data, e.g., centric representations [284] as well as containment-based representations [255]. An important generalization concerns the application of different visualization techniques for different degrees of detail. As an example, the Table Lens [207] displays a detailed textual representation for a user-defined focus region of a data table while the rest is shown only graphically (see Fig. 2.11). Another generalization concerns the

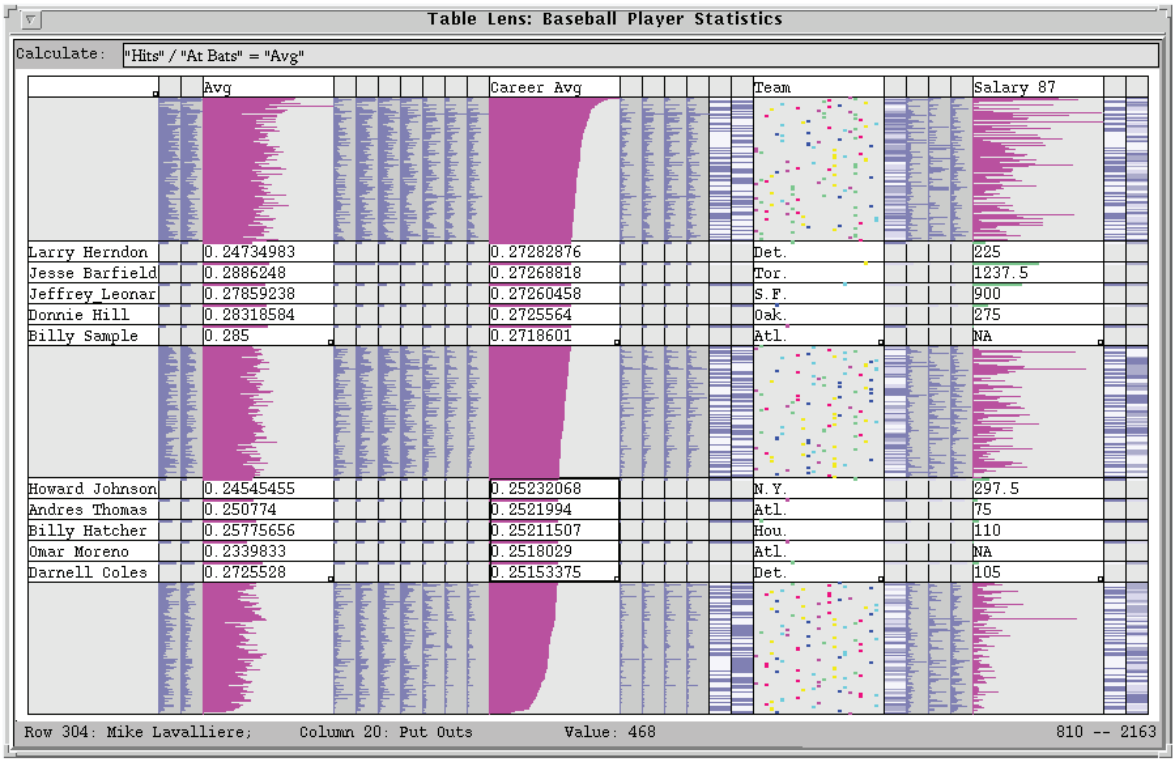


Figure 2.11: The Table Lens show a different degree of detail for the focus and the context [207].

definition of the degree of interest to incorporate data-driven instead of purely interactive approaches. World Mapper [56], for example, distorts a map in order to scale different territories by quantitative properties like the average income while preserving the topology between the territories.

Evaluations comparing distortion-oriented focus+context techniques against alternative options like zooming or overview+detail reveal mixed results [31, 190]. Positive results of distortion-oriented techniques have been reported for tasks like interactions on small screens [31], steering navigation [100], and calendar use [18]. For other tasks like interactive layout [99] and visual scanning [150], distortion was not the preferred option.

As another type of focus+context techniques, *in-place focus+context* employs visual cues instead of distortion to discriminate focus from context. A typical application is to highlight a certain subset of data by modifying visual attributes – mostly color or the size of glyphs. Inspired by the well-known depth-of-field effect in photography, Kosara et al. also proposed the use of frequency as visual cue [156]. Their semantic depth of field thus blurs objects to indicate them as context (see Fig. 2.12). A very similar approach has recently been proposed to lessen the visual structures of uncertain data [73].

Some in-place focus+context approaches modify the style and the level-of-detail for different degrees of interest. For example, Matkovic et al. describe different levels of detail for control widgets in process visualization [173]. Their design space trades off the size of widgets against the amount of shown information and also the precision of its visual representation. Novotny and Hauser also apply different visualization styles for representing outliers and

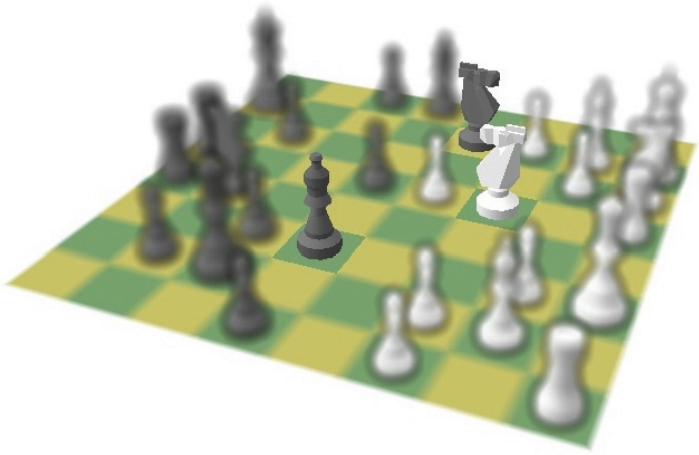


Figure 2.12: An example of semantic depth of field [152].

trends in parallel coordinates [194] (see Fig. 2.4). In this case, binning is used for the data considered as "trend" (see section 2.2.2) while "outliers" are drawn as poly-lines. In some sense, probes can also be regarded as in-place focus+context technique although this technique is closely related to multiple views as well. Butkiewicz et al., for example, draw iconic representations of multivariate views as detail information for certain spots of a geographical context [32].

As a conclusion, discriminating interesting from contextual information is agreed as a suitable and effective approach to achieve visual scalability in interactive visualization. As surveyed in this section, the design-space is huge which may be considered good or evil. Contradicting evaluation results when comparing different approaches suggest that the "ideal" technique is highly task-dependent and also dependent on the user skills, the environment, and on other parameters. Therefore, choosing the best approach for a particular design is still a non-trivial task.

2.5 Data Management and Parallelization

Most approaches discussed so far focus on visual aspects of scalability. In contrast, this section deals with computer-oriented issues of information and computational scalability with respect to data management and data processing. In scientific visualization, technical issues for handling large data have long been a major topic of research. In their introduction to this topic, McCormick and Ahrens identified four fundamental techniques to solve the large-data visualization problem in scientific visualization [176].

- Data streaming
- Task parallelism
- Pipeline parallelism
- Data parallelism

In information visualization, most approaches are still geared towards sequential CPU-based processing of comparatively small datasets which can entirely be kept in the main memory of common PCs. In the context of visual analytics, however, the importance of computational scalability has recently been increasing also for non-scientific visualization. This section surveys approaches to scalable data management and to parallel computing, focusing on their adoption by information visualization and visual analytics. After discussing data management for visualization in section 2.5.1, the survey of parallelism is structured by discriminating CPU-based approaches in section 2.5.2 and GPU-based approaches in section 2.5.3.

2.5.1 Data Management

Large data often exceeds the capacity of the main memory. Huge data even exceeds the capacity of local hard disks of a normal desktop PC and is typically stored on dedicated servers. According to the idea of memory hierarchies and caching [199], a common approach for both cases is to fetch data from the storage medium being next in size. This approach is often referred to as out-of-core computing as surveyed by Vitter [265]. Vitter distinguishes between *batched problems* and *online problems*. In the first case, all data items are processed by streaming them in blocks. In the latter case, the computation is done in response to a continuous series of query operations so that only a very small portion of the data needs to be retrieved in response to each query.

In *scientific visualization*, streaming structured data has long been a topic of research. Law et al. [162] describe a multi-threaded streaming architecture that automatically breaks data into pieces of a specified size and processes them within a pipeline of filters. Law et al. argue that their architecture supports any size problem on any size computer, at the expense of extra computational time. More recently, Cedilnik et al. describe an approach to remote large data visualization in the ParaView framework [35]. Their solution incorporates a data-parallel data server, a data-parallel rendering server, and a client controller. As a consequence, data parallelism is a necessary requirement in both cases which means that the data can be separated and pieces of data can be processed independently. While many algorithms for structured data in scientific visualization meet this requirement, many algorithms in information visualization do not.

This may be a reason why Vitter's notion of online problems better matches the needs of *information visualization* than batched problems. In fact, querying, sorting, and aggregating data items are key issues in many systems and precisely coincide with the core functionality provided by databases. Moreover, commercial databases and data warehouses are often the natural source of raw data. For these reasons, approaches that directly operate on top of databases are becoming increasingly popular in information visualization and especially visual analytics [146]. However, a distinction should be made between loose and tight integration between the visualization application and the database. Loose integration refers to approaches that retrieve data at specific points in time (e.g., at import) while relying on internal data handling mechanisms which are typically restricted to the main memory during the analysis. VisDB [144], Spotfire [1], and Advizor [124] are only a few examples of this approach which is fast for analyzing moderately sized datasets but does not scale to truly large data.

In contrast, analytics applications realizing a *tight integration* directly rely on the database for data management and data querying. As examples of a tight integration with database systems, front-ends for On-Line Analytical Processing (OLAP [39]) like Cognos [123] or the MicroStrategy Reporting Suite [181] let the user perform drill-down and roll-up operations

in huge data warehouses, but typically offer only static business graphics (see section 2.2.1). While OLAP optimizes query throughput by pre-computing aggregations, Harinarayan et al. [104] argue that carefully planning pre-computed views is crucial due to the large number of possible views. Tesone et al. propose smart aggregation to adjust the amount of data obtained from a database combining automatic measures and user-defined controls [245]. Doshi et al. describe asynchronous data pre-fetching to explore large datasets by adaptively selecting different pre-fetching strategies over time [58]. Wylie and Baumes present a component-based pipeline framework for the integration of algorithms from scientific and information visualization [279]. A key aspect of their work is a mechanism to use databases and database queries as the input to an on-demand data pipeline.

Especially with regard to a tight integration of database systems and visualization applications, the performance of the database is becoming an important issue for the usability of the interactive analysis. Most relational database systems are row-oriented, which means that the attributes of a data record are stored contiguously. While such an architecture optimizes the throughput performance when processing on-line transactions, it is less efficient for analytical queries accessing only a small subset of data attributes. Therefore, column-oriented storage is becoming increasingly popular for analytical tasks because only referenced attributes need to be fetched [242]. Comparing row- and column-oriented architectures, Harizopoulos et al. [105] conclude that column-oriented storage usually achieves higher performance. MonetDB [25] and kdb+ [125] are examples of column-oriented databases that also attempt to keep data in memory.

Based on kdb+, Chan et al. developed a client-server system for exploring massive time series [36]. 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.

Concluding, column-oriented in-memory databases can be an important step towards making interactive visual analysis of truly large data technically feasible. They allow for accessing the data in a controlled and standardized way, are able to handle issues concerning security and data consistency, and provide support for distribution and load balancing. All these issues are typically disregarded due to their complexity for proprietary data management facilities on top of which most visualization systems are built nowadays. However, research on how to efficiently incorporate such databases for visual analytics has just begun. A collaborative effort of the database community and the visualization community will be needed to design ways for an optimal interplay even in case of huge data.

2.5.2 CPU-Based Parallelism

Multi-core technology has become commonplace in today's CPUs and the number of cores will further increase in the future. As a consequence, utilizing CPU-based parallelism by means of multi-threading has become a major concern for all types of software, and it is a particularly important issue for demanding applications like interactive visualization of large data.

Parallelism and concurrency 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. Defining design patterns for particular problems has proven a good

approach to cope with this complexity. Schmidt et al. [218] describe 17 patterns for concurrent and networked objects, covering event handling, synchronization, and concurrency. Similarly, Mattson et al. [175] 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 [115] summarize the theory when programming for multiple processors and describe practical implementations for concurrent data structures.

While parallel algorithms and systems play an important role in scientific visualization, many techniques focus on exploiting data parallelism by parallelizing the processing of data blocks [162] (see section 2.5.1). For this purpose, most approaches specifically tune the internal representation of the data to maximize performance. In contrast, information visualization and visual analytics systems typically can not make as many assumptions about the data while offering the user many options to control the visualization pipeline. This may be a reason why these fields have paid little attention to multi-threading so far.

Among the few exceptions, Heer et al. [109] note that an important issue in implementing the Scheduler pattern is to handle concurrency. Their framework Prefuse [112] offers a scheduler mechanism to execute costly computations in a separate thread, e.g., to drive animations or incremental layout computations. The XmdvTool uses multi-threading for asynchronous data pre-fetching [58]. The visualization system Improvise [272] implements asynchronous displays based on retarding worker threads to allocate as much resources as necessary to the user interface thread (called *throttling* [275]).

A common problem of applying multi-threading in a general way is the increased system complexity and the usually higher implementation costs [163]. Heer and Bostock note that addressing this problem was one motivation for them to design a declarative language for information visualization [111]. Their language seeks to separate the specification of visualizations from details of the execution in order to simplify the development and to support optimizations like parallelized execution.

As a conclusion, research on utilizing CPU-based parallelism in information visualization and visual analytics is still in its infancy. Most systems apply multi-threading only for isolated aspects – if at all – and many non-trivial details about multi-threading have been left unpublished. This situation was a motivation for proposing a generic multi-threaded architecture (see chapter 3).

2.5.3 GPU-Based Parallelism

Driven mainly by the computer games industry, Graphics Processing Units (GPUs) have seen a tremendous evolution in the past 15 years. Showing a growth-rate of 2.5 – 3.0 times a year, the performance of GPUs has been increasing faster than Moore's Law for CPUs [198]. Today, graphics cards in common PCs are equipped with GPUs of 512 parallel cores and more than a gigabyte of highly optimized memory.

While GPUs have mostly been designed for realistic real-time rendering of large scenes, *general-purpose GPU computing* (GPGPU) [248] attempts to make use of the stream processing capabilities of GPUs for general computation and algorithms. A key issue of GPGPU is to map high-level data structures to the graphics primitives in terms of which GPUs are programmed [164]. GPGPU libraries like NVidias CUDA, ATIs Stream Computing SDK, and the Brook library [29] are very helpful in this respect.

GPUs have also been used for visualization, but mostly in the context of scientific vi-

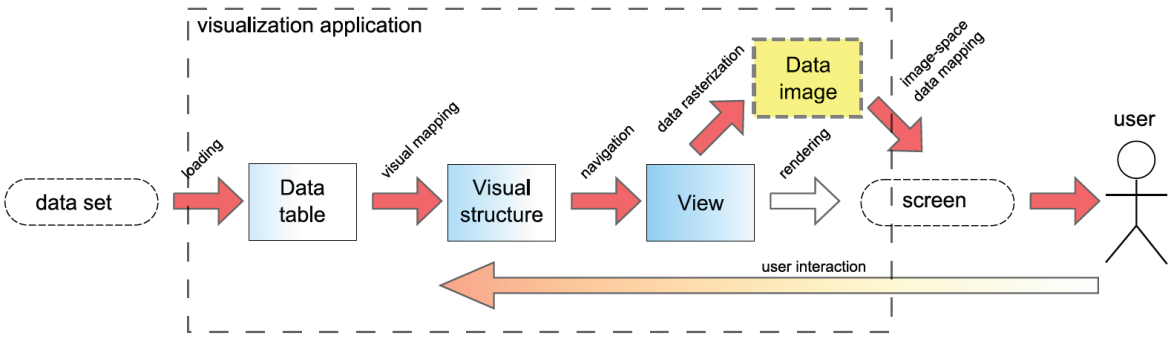


Figure 2.13: The visualization pipeline as extended by an image-step [178].

sualization. Weiskopf summarized approaches for GPU-based scientific visualization [276]. An adoption of the GPU by information visualization has been much slower. Existing work often discusses the utilization of GPUs rather as an implementation detail than as a part of the contribution. Johansson et al., for example, use a GPU pixel shader to render parallel coordinates [135]. Elmqvist et al. apply a fixed GPU-shader architecture to draw glyphs in a graph visualization system [66].

Other work investigated the use of GPUs for force-directed node placement. Frishman and Tal present an algorithm for force directed graph layout on the GPU [80]. Their approach computes a balanced partitioning of a general graph to allow for a data parallel programming model, but it does not scale to weighted complete graphs and uses the CPU for initial node placement. Glimmer [126] is a GPU-based approach to Multi-Dimensional Scaling that runs all stages of the algorithm on the GPU. Glimmer recursively applies a parallel, force-based subsystem to combine and refine hierarchically organized input data.

There have also been first approaches to utilize GPUs in information visualization in a more general way. Fekete and Plaisant [71] investigated methods based on hardware acceleration to interactively visualize a million data items in scatterplots and tree map visualizations. Besides rendering performance, non-standard visual attribute mappings support perception, and appropriate interaction methods are integrated. Florek and Novotny [76] utilize graphics hardware to improve the rendering performance of scatterplots and parallel coordinates. More recently, McDonnel and Elmqvist describe an image-space step as refinement to the information visualization pipeline for an implementation using GPU shaders [178] (see Fig. 2.13). Their strategy is to sample multivariate data in the resolution of the current view.

Concluding, GPUs are likely to play an increasingly important role in information visualization and visual analytics in the near future. However, McDonnel and Elmqvist conjecture that a wide-spread adoption of GPUs is currently hindered by a conceptual mismatch between the geometrically-related basic data types of GPUs and more high-level and abstract datasets such as graphs, trees, and free text [178]. Heer and Bostock note that optimizations like GPU processing lead to increasingly complex APIs, which impose an additional burden on developers [111]. Therefore, finding a good balance between complexity and efficiency will be a key to success.

2.6 Approaches Adressing Other Scalability Issues

As discussed in chapter 1, scalability in visual analysis is a very broad topic that goes far beyond the sheer size of analyzed data. In accordance with the topic of this thesis, this state of the art report focused on scalability with respect to large data. However, this final section briefly summarizes approaches to address other scalability issues.

Concerning *display scalability*, the limited resolution of small screens makes them a particularly challenging field of research. As one example, Büring et al. proposed interaction techniques to support scatterplots of multiple thousand items on displays of smart phones [31]. Another approach are facet-based interfaces like FacetMap [233] and FaThumb [139].

At the other end of the spectrum, Yost and North investigated the perceptual scalability of visualizations to very large displays [287]. Their results suggest that encoding is more important on a smaller display while spatial grouping is more important on a larger display. Multi-projector displays are one technique to realize large screens in practice, but a seamless combination is challenging [216, 215].

Human scalability summarizes aspects of collaboration, which is in some cases related to large displays. For example, Isenberg et al. studied co-located collaborative visual analysis around a tabletop display [129]. However, co-located collaboration of small groups is just one case of social interaction. Heer and Agrawala emphasize that the most appropriate collaboration mechanisms to support sense making are not immediately clear [110], and they discuss design considerations for asynchronous collaboration.

Web-based approaches are becoming increasingly popular to support collaboration on a large scale. Many Eyes [264], for example, is a public web site where users may upload data, create interactive visualizations, and discuss findings. Being targeted to the public, such approaches face additional challenges as compared to designing visualizations for trained expert users. Respective techniques have sometimes been termed casual information visualization. Examples of casual information visualization also include automated generation of visualizations [169], visual story telling [221], and various social applications like installations in museums [117], exploration of the user's own history [13], and explorations of online conversations [55].

Involving textual data, relational data, and multimedia data, online conversations also illustrate that visualization has to deal with increasingly *heterogeneous data*. Only a few approaches have been proposed that adress a joint analysis of heterogenous data. As one example, VisLinks [46] reveal relationships between multiple visualizations. Ivanov et al. describe an example to integrate data from video cameras and motion sensors [130].

Dealing with scalability issues typically also increases the complexity of writing respective analytics software. In addition, software systems are required to be modular and highly configurable while the development should be cost-efficient and fast. In order to achieve these complementary goals of *software scalability*, design patterns for information visualization describe reusable building blocks [109], and various libraries and frameworks support the development process at different levels of abstraction [111, 74].

Chapter 3

A Multi-Threading Visualization Architecture

As motivated in chapter 1, interactive visualization is particularly suitable for the exploration of unknown data. Explorative tasks are different from presentation tasks in that they require frequent changes of the view on the data. Therefore, a smooth and efficient exploration requires that the ensemble of analytical, visual, and interaction methods has to generate results in a timely manner (within 50 – 100 ms [226, 236]). However, even moderately sized data can pose computational challenges. Computing a graph layout of a few hundred nodes or rendering a dataset with a million 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) scalability problems like 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 [67]. This is in particular true for an exploratory visual analysis which typically involves examining multiple 'what if' scenarios [236]. 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. A support of continuous interaction is thus considerably more challenging with respect to the scalability to large 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. This significantly limits the scalability

to large data.

Therefore, some systems (e.g., Improvise [273]) parallelize these tasks using *multi-threading*. While multi-threading is desirable, many potential pitfalls have not sufficiently been addressed in the context of interactive visualization so far (see section 2.5.2). 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.

This chapter describes a generic visualization architecture that intends to increase the *computational scalability* of visual analysis systems based on multi-treading. In particular, the architecture 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 [226]
- Provide as much visual feedback as possible
- Scale to datasets 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 (e.g., maximizing speed vs. maximizing the amount of feedback), particular design choices are explicitly outlined and discussed. This architecture has been shaped based on experiences in implementing several visualization systems and tools, including SimVis (C++) [52], Visplore (C++), CGV (Java) [251], and VisAxes (C#) [250].

The next section relates the proposed architecture to other work. Section 3.2 describes our architecture, including details related to multi-threading. We present a quantitative evaluation based on the system Visplore in section 3.3. This chapter closes with a discussion about design choices, further instantiations of our architecture, and ideas for future work in section 3.4.

3.1 Related Work

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

3.1.1 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 an elastic

rectangle during dynamic query operations, whereas the query evaluation is triggered only after releasing the mouse button.

Tanin et al. [243] describe optimizations to dynamic queries which are an important type of continuous interaction in many systems. 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 [70], 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.

Chapter 2 surveyed a variety of approaches to speed up rendering based on data removal or aggregation. However, some of these methods involve costly computations (e.g., clustering) which may cause a temporary loss of application responsiveness in case of large data. Moreover, most approaches imply a loss of details, which is not always acceptable.

3.1.2 Concurrency and Parallel Programming

Many real-time graphics applications (e.g., games) exploit the parallelism of modern Graphics Processing Units (GPUs) to achieve interactivity when transforming geometric or volumetric data into images. GPUs have also begun to attract attention in visual analysis [71, 178] (see section 2.5.3), but transferring all steps of the visualization pipeline to the GPU is not always possible.

Chan et al. [36] delegate data queries to eight multi-processor database servers for exploring massive time series. However, constraints limit the distance that a user is allowed to travel per exploration step. It remains unclear how far such large-scale architectures 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 [70]. 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 during continuous interaction in general.

Parallelism and concurrency in a general sense involve numerous highly non-trivial issues. 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 [163]. 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 been termed the *Multicore's Programmability Gap* [180].

Defining design patterns for particular problems has proven a good approach to cope with this complexity [218, 175, 115]. Many patterns are applicable to systems for visual data analysis and some patterns are even related to the architecture as proposed in this chapter (e.g., the *Active Object* design pattern [218]). However, the scope of most patterns is very general and neither addresses the requirements regarding responses to user interaction nor visualization aspects.

3.1.3 Multi-Threading in Interactive Visualization

While multi-threading is frequently used in *scientific visualization*, many techniques focus on exploiting data parallelism (see section 2.5). On a task level, computations in SciRun [137] are multi-threaded and do not block the GUI, but are typically not designed for early cancellation due to new input. The system ParaView [35] 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 [33]. For this purpose, most approaches specifically tune the internal representation of the data to maximize performance.

As discussed in section 2.5, information visualization tools typically can not make as many assumptions about the data. This may be a reason why little attention has been paid to multi-threading in information visualization literature so far. The few exceptions typically provide little information concerning communication and synchronization aspects (e.g., Heer et al. [109]), or use multi-threading only for asynchronous data pre-fetching [58]. A review of open source visualization software shows that The InfoVis Toolkit [70], Processing [81], and Mondrian [246] do not employ multi-threading at all.

Most closely related to the proposed architecture, the visualization system Improvise [272, 273] uses patterns for coordinating multiple views which are related to semantic layers (see section 3.2.2). Asynchronous displays in Improvise are based on retarding worker threads to allocate as much resources as necessary to the user interface thread [275]. 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 [273].

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 interactive information visualization tools, which combine responsiveness and rich visual feedback even during continuous user interactions.

3.2 Multi-Threading Visualization Architecture

We first provide an overview of the proposed architecture before discussing its details in sections 3.2.1 and 3.2.2. The architecture builds on the separation of the main application thread and visualization threads (see Fig. 3.1). 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, all user requests arrive in the main application thread, while each view has its own visualization thread.

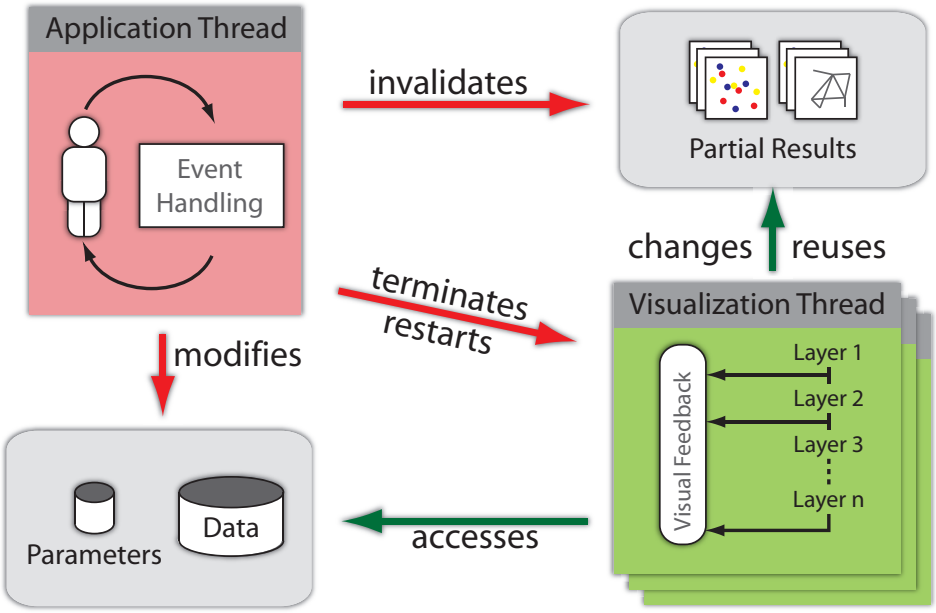


Figure 3.1: Overview of the architecture. It shows involved threads and data, how threads access this data, and how the application thread controls the visualization threads.

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 between the application thread and the visualization thread. 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. Event handlers invalidate layers so as to maximize layer reuse and thus to accelerate updates. 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.

3.2.1 Early Thread Termination

Our approach to support continuous interaction even for large data 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 [218], 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 [41] 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 3.4). 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 [226, 236], 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 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 Fig. 3.2) 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 Fig. 3.2) 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,

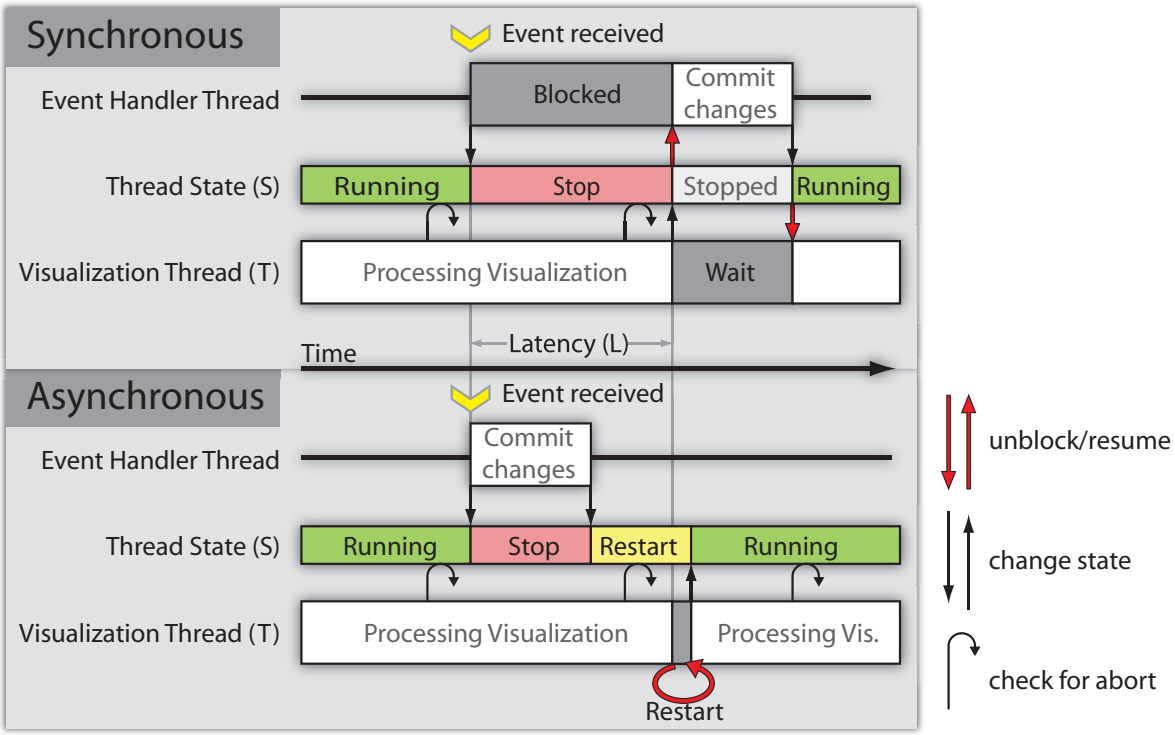


Figure 3.2: Comparison of synchronous and asynchronous event handling. Threads communicate by changing the thread state S.

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 3.2.2 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. Fig. 3.2 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. Fig. 3.2 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 chapter, 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).

3.2.2 Layered Visualization

As explained in section 3.2.1, identifying and reusing partial results during the execution of the visualization thread is necessary to avoid redundant computations. 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,

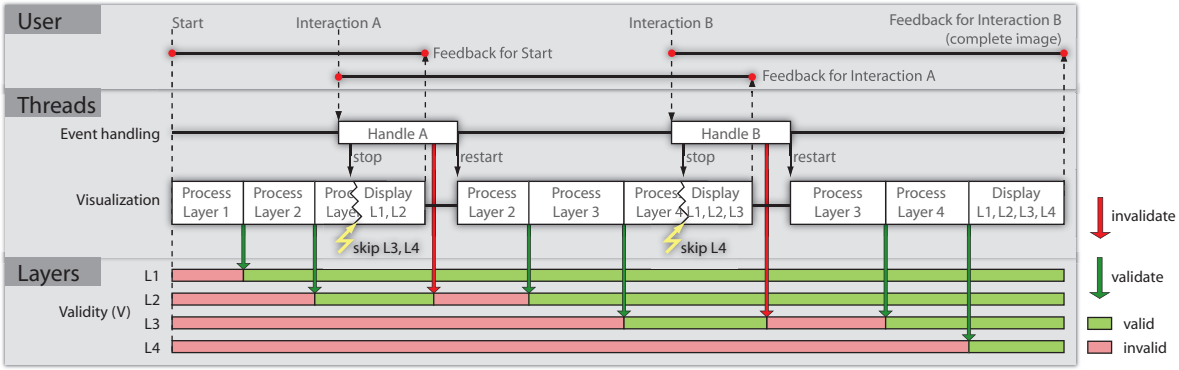


Figure 3.3: Caching and early feedback of layers. Two user events (handled synchronously) interrupt the computation and invalidate layers. Visual feedback is provided on thread termination.

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 [273]. 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 scatterplots 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. Ellis et. al. [63] have shown that statistical sampling is an effective way of data reduction.

- *Level-of-detail (LoD) layers* provide visual representations of the same data with different complexity and rendering cost. The processing order is determined by increasing effort. 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 [23]. 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 [194]. 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 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 [41], 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 [162]. 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. Fig. 3.3 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 visualizations (scatterplots, 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.

3.3 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 [51], which also implements the proposed architecture (see section 3.4). 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 Visplore, a system for visual exploration. Visplore provides more than 10 different visualizations (e.g., the approaches proposed in the chapters 5, 6, and 7). 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 focus of this evaluation is to demonstrate the possibilities of our architecture. An application of Visplore is described in chapter 7.

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

- *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.

	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 3.1: Results for example 1: restricting a slider

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 3.1 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 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 scatterplot and highlight the selected data items in a linked parallel coordinates view showing 5 axes (see Fig. 3.4). 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

	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 3.2: Results for example 2: linked ad-hoc selection

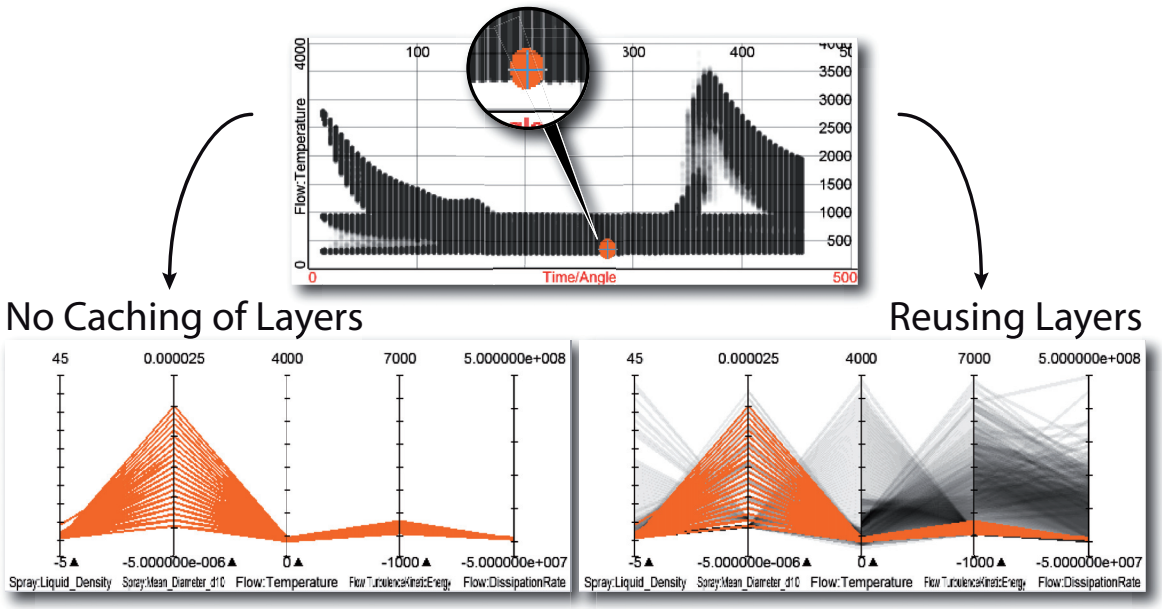


Figure 3.4: Example 2: comparison of an ad-hoc selection of entries beneath the mouse cursor in a multiple view setup for two implementations of parallel coordinates. The response time is equally low in both cases, but the amount of detail is much higher when caching and reusing layers.

pipeline in the application thread

- *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 3.2 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 Fig. 3.4 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 Fig. 3.4. 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, the architecture also scales with respect to a large number of views. Informal evidence can be found in the application example of chapter 7, as well as in several publications related to the systems implementing the architecture (for example [52]).

3.4 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 (e.g., pivoted data as in chapter 5). 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.

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

Another limitation with regard to computational scalability concerns the *number of involved threads*. The architecture utilizes one thread for each view plus the main thread of the application, which results in less than ten threads for most multi-view systems. While this roughly corresponds to (or even exceeds) the number of cores in current desktop PCs, computers will have much more cores in the future. Since increasing the number of views for the sake of enhancing parallelism is obviously no option, making use of all available cores will become increasingly challenging. In this context, it is reasonable to utilize additional threads for other tasks than visualization, e.g., the computation of derived data or the evaluation of queries.

As another potential solution, it may seem reasonable to spawn a new visualization thread for each asynchronous event (synchronous changes must wait for thread termination anyway). In the systems implementing the architecture, we decided against this option, because our practice has shown that gains are small on current computers 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. However, on computers having much more cores, utilizing multiple threads per visualization could become a reasonable option.

The proposed architecture has been implemented in several systems besides Visplore (see section 3.3). The *SimVis* visualization framework [52] is mainly used in the context of 3D or 4D simulation data (see also chapter 4). 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 scatterplots or time series visualizations [186] 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 [162] 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 [251]. 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* [250], 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 dataset 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 chapter, 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.

3.5 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 even in case of large data. Realizing this in practice is difficult, however, as it requires parallelism of application tasks which involves many non-trivial details.

This chapter proposed a generic architecture to support continuous interaction and to help avoiding pitfalls. Being inherently multi-threaded, the architecture improves the computational scalability of tools for visual analysis. As illustrated by the evaluation, the architecture also scales with respect to large data sizes. 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.

Chapter 4

Focus+Context Visualization with 2D/3D Scatterplots

Simple *2D scatterplots* are a very old and well-known visualization method for unstructured data. The two-dimensional display can be easily overviewed and understood, and lends itself well to direct interaction with two-dimensional input devices (e.g., the mouse).

This simplicity and ease of use hides a number of problems, however. In a large dataset, many points may be plotted onto the same pixel, without the user being able to tell how many. This makes it impossible to judge the true distribution of data from a scatterplot, thus limiting the visual scalability. For this reason, some approaches modulate the intensity of the color to convey the point densities in 2D visualizations [214].

As another problem of 2D scatterplots, it is also hard to find structures that exist in more than two dimensions. This has prompted various extensions of 2D scatterplots to deal with higher-dimensional data. A scatterplot matrix [43], for example, lays out scatterplots for all pairs of dimensions as a matrix so that plots within one column share a common X-axis, and plots within one row share a common Y-axis. Prosection views [86] rely on interaction to restrict the projected points to user-defined ranges in data dimensions which are not shown by the scatterplot. This idea has also been combined with the layout of a scatterplot matrix [258]. While these extensions are definitely very useful, it is still not intuitive and requires some training to develop a feeling of structures which involve more than two dimensions.

3D scatterplots solve a part of the limitations of 2D scatterplots. There is an additional dimension in which structures can be separated. Moreover, less data will in general be projected onto the same position in 3D space than in 2D space which reduces the overplotting problem to some degree. Several variants of 3D scatterplots have been proposed which can be distinguished as point-based or volume-based. Point-based 3D scatterplots draw a point for each data item. Several statistics and math packages like SPSS, R, and MatLab [172] implement this approach. In contrast, volume-based 3D scatterplots employ binning to generate a volume that represents the densities of the data. Volume rendering is applied to generate the visualization [15, 157, 209]. The rendering performance of volume-based approaches depends on the resolution of the volume rather than on the size of the data which is an advantage in the case of many data items. On the other hand, point-based approaches enable a continuous mapping from data space to scatterplot space which avoids the aliasing problems caused by binning.

In general, 3D has many disadvantages compared to 2D [232]. Interaction is harder in 3D,

and the projection of a 3D plot onto a 2D output device incurs a loss of information. This loss manifests as occlusion and causes difficulties in making precise length judgments because of perspective foreshortening [252]. Furthermore, most implementations lack sufficient depth cues, which makes a judgment of the three-dimensional structure of the points very difficult.

The *contributions* of this chapter intend to alleviate some problems of 3D scatterplots and to enhance their visual scalability for large data. First, we introduce halos and depth-dependent point size as novel depth cues in the context of 3D scatterplots, and we use histograms as technique to highlight the point distribution and density inside and outside a user-defined spatial focus region. Second, we propose a linked setting of the 3D scatterplot with three 2D views, which combines the advantages of both (see Fig. 4.1). Interaction is done in 2D, with the results shown in 3D. Third, we present a case study that illustrates the usefulness of the combined approach and the novel techniques.

The combined 2D/3D scatterplots were implemented as part of SimVis [52], a system that uses linked views [11] extensively for the display of high-dimensional data from computational flow dynamics (CFD) simulations. The 3D scatterplot is point-based rather than volume-based in order to avoid aliasing artifacts and ambiguities with respect to the degree of interest (see section 4.2). All parts of the approach exploit the capabilities of modern graphics hardware to provide interactive frame rates even for large datasets.

4.1 Extending 3D Scatterplots

This section proposes various extensions to standard 3D scatterplots. The extensions address common problems like a compromised perception of depth and point density.

4.1.1 Improving Depth Perception

A common problem of conventional 3D scatterplots is the loss of depth information after the projection onto the 2D output device. Interaction methods like changing the viewpoint temporarily compensate for this shortcoming due to the effect of motion parallax, i.e., the different movement speeds of the displayed points, but are not applicable to still images. To address this problem, our approach employs size and color to represent the distance from the viewpoint for each point and halos help to outline the shape more clearly (see Fig. 4.2).

Size is one of the primary natural depth cues [93], because the human brain is used to the fact that distant objects appear smaller, as they occupy less space on the retina of the eye. As an independent visual dimension (like color and position), size is also well-suited for representing depth in 3D scatterplots. Decreasing the point-size with increasing distance from the viewer enhances depth perception considerably. The size can either drop off reciprocally with the distance – mimicking a perspective projection in a mathematically correct way – or decrease linearly from a maximum to a minimum inside the 3D scatterplot, which permits a better discrimination in point size of the distant parts (which we found more useful in the context of our application).

Although varying the point size obtains a convincing depth impression for sparse areas, this effect levels off as soon as clusters of similarly colored points cover large, almost monochromatic spots on the output device, where single data entries can not be discriminated any more. In this case, *halos* [128] can help to outline the shape of single points. As a technique known from painting, accentuating the outline can help to intuitively indicate the presence of depth discontinuities between contiguous elements in a projection. This comes

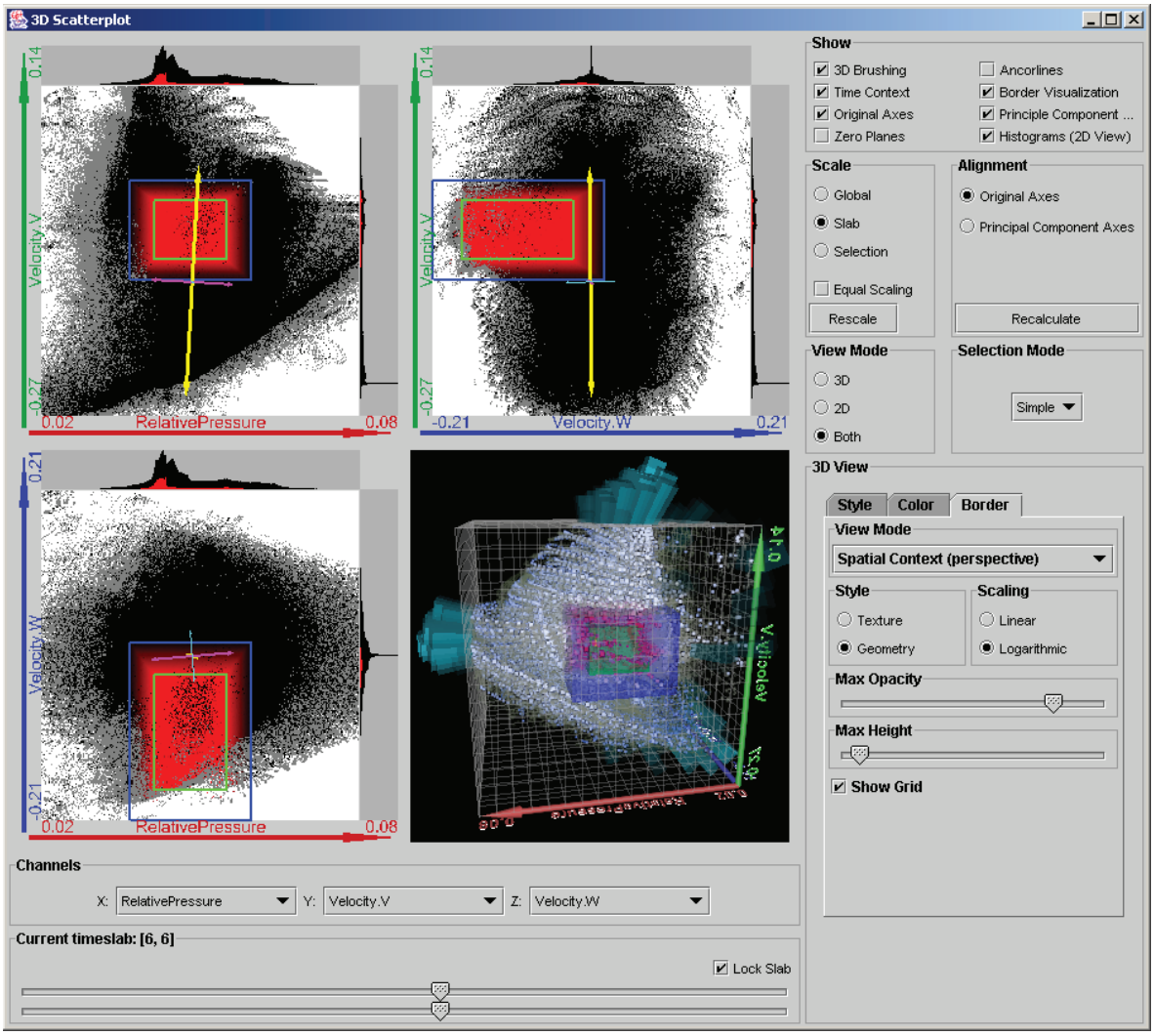


Figure 4.1: The combined 2D and 3D scatterplot with user interface.

at the cost of a slightly increased amount of occlusion, which we found hardly disturbing, since only the foremost front of a big cluster is visible anyway. Technically, each point is surrounded by a thin, semi-transparent circle of the same hue as the original point-color but with a much lower brightness. As illustrated in the middle of Fig. 4.2, varying the point-size in combination with halos conveys a three-dimensional impression and permits to make out single entries even in dense areas – unlike in the left image, where neither depth cues nor halos are used.

Apart from size, depth can also be mapped to *color*. Altering the hue and contrast of scene elements that are farther away from the viewer is a technique known from painting and rendering and is a frequently used sort of depth cueing [91]. Due to characteristics of the human visual system, bright and warm colors (like red or yellow) are suited to indicate proximity to the viewer, while dark and cold colors (blue or grey) are intuitively associated with depth. This optical phenomenon is referred to as chromo-stereoscopy [253].

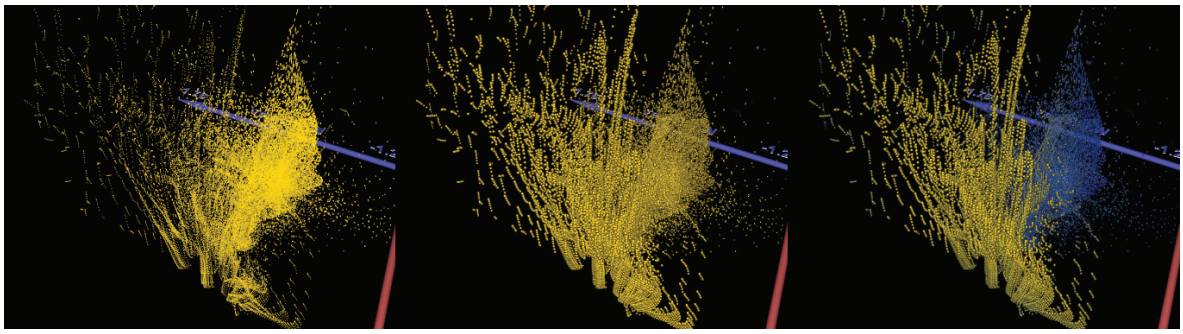


Figure 4.2: Improving depth perception. Left: No depth cues are used; Middle: Depth is indicated with point size and halos are used to ease the discrimination of single points; Right: Depth cueing using both color and point size, as well as halos.

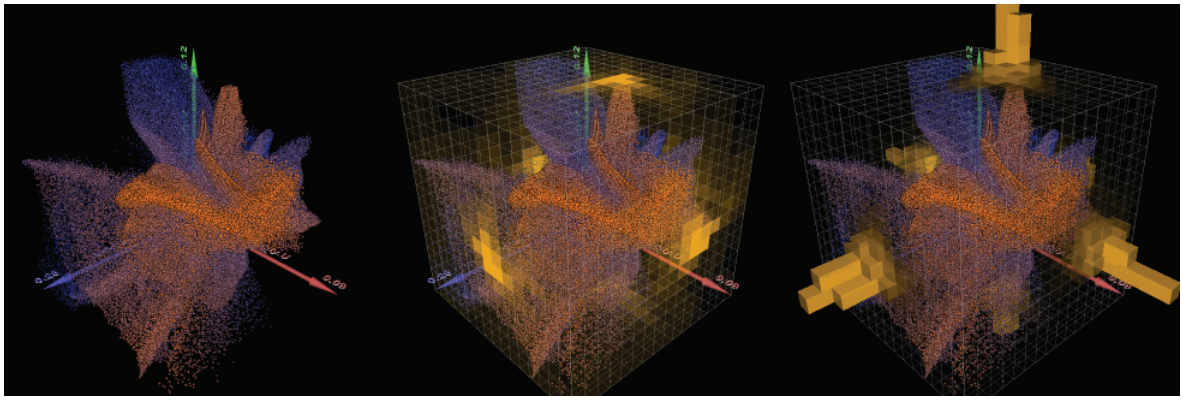


Figure 4.3: Representing the point density with transparent 2D histograms: The density is not clearly recognizable in the left image. In the middle image, a texture-based representation uses opacity only, while the right image illustrates mapping density to opacity as well as to the height of axis-aligned bins.

Depth cueing with point size, color and halos can be done independently or in combination, the latter yielding the most convincing depth perception (see Fig. 4.2, right). Employing dedicated graphics hardware for modifying point size, color, and for generating halos helps to achieve interactive frame rates for a large number of points. Vertex programs have proven useful for these tasks if the 3D scatterplot is point-based (as in our case), and stored as a set of independent vertices. Hardware-based fogging is another option, yet applies to color only and provides less flexibility than vertex programs. Our implementation uses one vertex program for all tasks, since the same interpolation factor can be applied to both point size and color, which is efficient and guarantees consistency.

4.1.2 Representing Point Density

The approximate point density at a certain position is important information when analyzing characteristics of a dataset. However, scatterplots (2D and 3D) represent *density* in a satis-

factory way only as long as hardly any points are projected to the same pixels. Adapting the scaling does not solve the problem of identical data in general and comes at the cost of losing overview. Our approach incorporates two-dimensional histograms at each border plane of the cube where the 3D scatterplot is drawn. Each rendered data point (lying inside the cube) is orthographically projected to each border plane. The 2D distributions are discretized by equally sized squares (called bins) in accordance with the desired resolution. A user-definable scaling is applied to the bin-counts before any further visualization. Linear scaling highlights peak densities while logarithmic scaling permits to discriminate even minor differences in sparse areas.

Our approach supports two different visual representations of the 2D histograms. A *geometric representation* draws one axis-aligned cuboid per bin (Fig. 4.3, right image), and the height reflects the respectively scaled bin-count. A *texture-based flat visualization* uses one quadratic texture per border plane matching the resolution of the binning with nearest-neighbor filtering (Fig. 4.3, middle image). In both cases, the scaled bin values are mapped to the opacities of the cuboids or texels, respectively, with a user-definable maximum opacity. Additional color coding is possible, but may cause problems due to too much visual complexity. Concrete implementations must take into account that transparent geometry requires a view-dependent back-to-front rendering order.

Although both visualization methods can not solve the problem of unrecognisable point densities inside the scatterplot itself and allow for rather approximate assessments only, they provide very useful information and contribute to a better understanding of the data. Section 4.2.2 adapts this technique for 2D scatterplots, where we face similar problems in representing the point density.

4.1.3 Spatial Context Information

When analyzing large datasets, zooming into the data helps to focus on local features like clusters. In our approach, the user may define a cubic cutout of the scatterplot (referred to as spatial focus). The points outside this cube are not rendered, but may still provide useful context information (the spatial context). We discuss a visualization of this context similar to the way as we deal with point densities in section 4.1.2: After scaling the data as currently chosen, all points of the spatial context are projected onto the border planes of the cube as described below and binned according to the desired resolution. We perform either linear or logarithmic scaling before the results are mapped to opacities and displayed using a texture or geometry-based transparent representation.

We propose a *perspective projection* towards the center of the cube of all points, which are outside the spatial focus (see Fig. 4.4 left for a 2D sketch). Each border plane serves as view plane for all points of the spatial context seen from the cube center using a quadratic view frustum with a field of view of 90 degrees. This approach strongly resembles the way cube maps [95] are generated in the field of real-time rendering (for instance for environment mapping) and it actually serves a similar purpose.

For binning, the angle of the field of view (rather than the view plane) is equally subdivided in order to make sure that each bin covers an equal amount of space. Each bin represents the number of points of the spatial context in a certain direction. In order to intuitively indicate the direction captured by one bin, we propose to use projection-aligned bars as geometric representation (Fig. 4.5, left). Using non axis-aligned geometry for a perspective projection also renders it easier for the user to visually distinguish between the various visualization

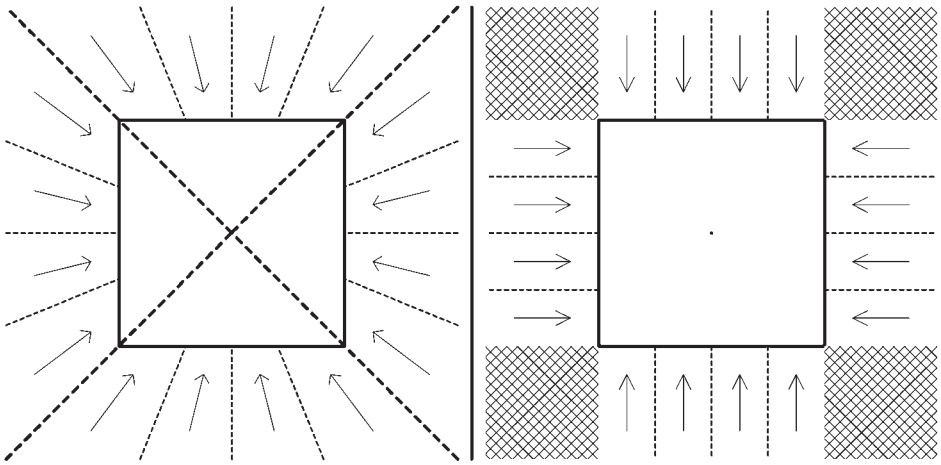


Figure 4.4: Projecting the spatial context perspectively (left) or orthographically (right). The hatched space is not captured by the orthographic projection. Note the unequal binning when equally subdividing the angle instead of the plane with the perspective projection.

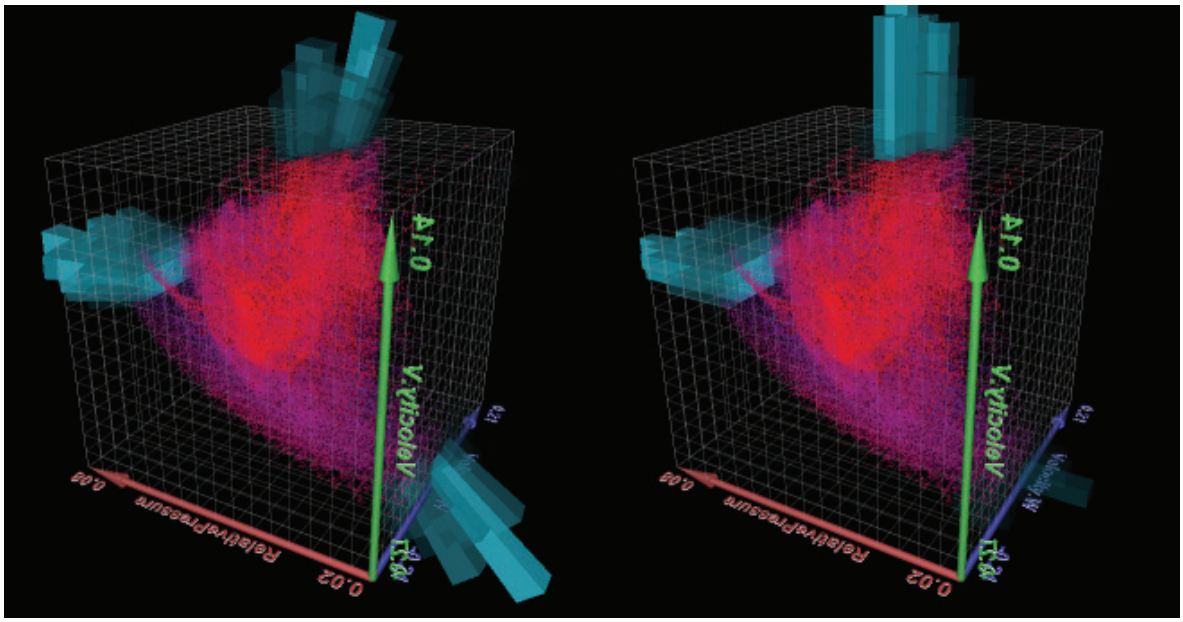


Figure 4.5: Representing the spatial context: The perspective projection (left, drawn using projection-aligned transparent bars) captures the whole spatial context, while the orthographic projection (right) omits parts of it (the lobe in the bottom right corner for example).

modes. The scaled bin values are again mapped to the heights of the bars. It may possibly seem more intuitive to place the bars of the perspective projection on a sphere, but we have decided against this representation as it differs from the shape of the actual spatial focus and leaves a gap in between. A texture-based representation is somewhat problematic, since the unequally sized bins mismatch the equally sized texels. A simple solution is to draw one

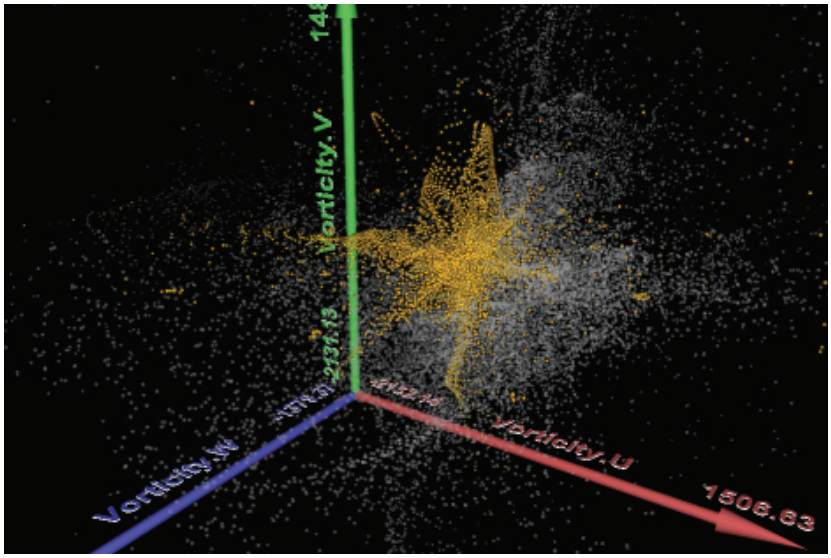


Figure 4.6: Displaying the temporal context as grey points.

correctly sized flat quad per bin instead.

Another option is to use an *orthographic projection* instead of a perspective one. However, without extensions, this is applicable to only such data entries, which are outside the spatial focus in not more than one considered dimension; These points are projected onto the nearest border plane as illustrated in the right of Fig. 4.4. The result can be equally binned and visualized as texture or using axis-aligned cuboids (Fig. 4.5, right). Although this method is simpler, we still favor the perspective projection, since it captures the entier spatial context. Neglecting the parts of the context which are outside the focus in two or three dimensions can be misleading.

4.1.4 Temporal Focus – Context Discrimination

The SimVis system [54], which the 3D scatterplot is part of, is designed for analyzing flow simulation results over time, thus the dimension *time* is essential in all views and should always remain within the attention of the user. As an alternative to mapping time to the axes of the scatterplot just like any other data attribute, the view provides a range slider as a means of specifying a certain time span of interest – called temporal focus – within the overall duration of the simulation. Only this focus is subject to any kind of interaction and spatial context visualization. We refer to this kind of focus – context discrimination as temporal focus – context discrimination. However, a generalization to other dimensions than time would be straightforward and would resemble the idea of prosection views [86].

The temporal context is optionally rendered as semi-transparent grey points – analogously to the visualization in other SimVis views [52] and clearly discernible from the opaquely colored points of the temporal focus (see Fig. 4.6). Since the rendering order is important for all transparent objects, we suggest drawing the entire temporal focus before the temporal context with the depth-test enabled. This ensures that the focus remains visible even behind the context. The user can define the point sizes separately for focus and context permitting

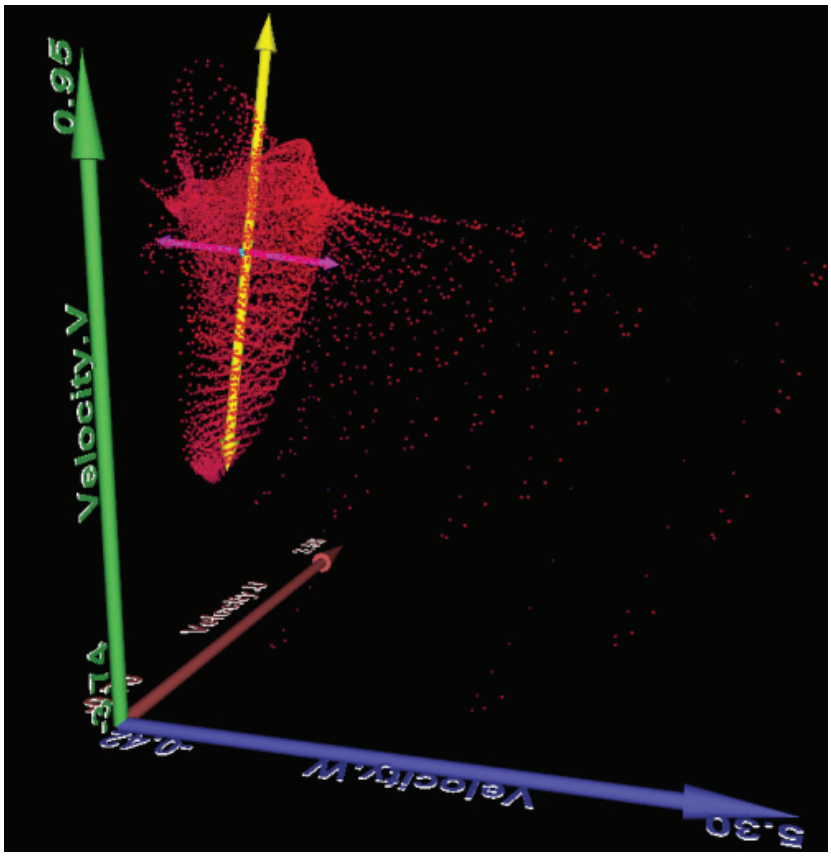


Figure 4.7: Displaying the principle components of the displayed data: The first axis (yellow) indicates the direction of the likeliest correlation, the second axis (magenta) is less than half the size and the third axis (cyan) is hardly visible, thus the data seems to be actually correlated.

to highlight one or the other. Halos and depth-dependent point sizes as techniques for depth cueing (see 4.1.1) are applicable to the context as well.

4.1.5 Displaying Principle Component Axes

An important reason for drawing scatterplots is to relate two or three dimensions in order to visually detect potential correlations. A mathematical way to deal with correlations between dimensions is to perform a *Principle Component Analysis* [138]. Transforming potentially correlated variables of an n-dimensional data space into n uncorrelated variables with decreasing variability yields an orthogonal basis of the data space (see section 2.3). The axes of this basis are ordered by the amount of variance the data shows in the respective direction. This is valuable information when exploring the characteristics of the dataset and is thus (optionally) visualized. The principle components are calculated for the spatial and temporal focus of the three dimensions mapped to the axes of the scatterplot in scatterplot space and displayed as 3D arrows (Fig. 4.7). We use the mean values of all considered data entries as origin of the obtained coordinate system and scale the displayed arrows in accordance to the

absolute values of the respective eigenvalues, whose computation is an intermediate step in the overall calculation of the principle components.

4.2 Interactively Linking 2D and 3D Scatterplots

The main advantage of 3D scatterplots is that one more dimension is simultaneously displayed. However, 2D scatterplots are much more widely used and thus much more familiar to the majority of users. Besides, the two dimensions of the mouse as standard input device match the dimensionality of 2D views, which makes interaction with the data much easier and more intuitive. This section proposes a combination of 2D and 3D scatterplots and discusses linking and brushing [30] in this context.

4.2.1 Assisting 3D Viewing with 2D Scatterplots

A common way to complement 3D viewing with 2D views in commercial modeling applications is to add three 2D views, each showing an orthographic projection of the scenery for the X-, Y- and Z-axis, respectively. Applying this approach to 3D scatterplots yields a spreadsheet forming a simple 2D scatterplot matrix [43]. This matrix displays every combination of two data dimensions mapped to the three axes of the 3D scatterplot (see Fig. 4.8). The arrangement is important, as neighboring views assign the same dimension to the common edge to ease comparisons. In order to attain a consistent multi-viewing, changes to any relevant parameter (e.g., axis-mapping, scaling and so on) take immediate effect in all views. An in-depth discussion about using linked multiple views in a spreadsheet setting is provided by Chi et al. [42].

Apart from providing more familiar 2D scatterplots as a purpose on its own, a significant advantage of the combined approach is the ability to easily define *axis-aligned 3D brushes* in a 2D environment. Although conceptually identical to 2D brushes, it is important that the brush concept matches the view layout: Using the feature specification as proposed by Doleisch et al. [52], a brush defined within the proposed setup is a logical AND-combination of selections on all three dimensions of the dataset which are currently mapped to the axes. Such a brush (referred to as simple brush) represents an axis-aligned cuboid in 3D and a rectangle in 2D. Due to the view layout, the boundaries of a brush are collinear in neighboring 2D views. Simple brushes are created by dragging the mouse in any 2D view, where the two dimensions mapped by this view are constrained according to the user input, while the third (hidden) dimension is initially defined by the size of its spatial focus. After creation, simple brushes can be moved and resized in any 2D view which takes instantaneous effect in all views. We only support axis-aligned brushes, because brushes of this shape do not implicate any correlation between the dimensions [52]. However, the user can define arbitrarily complex composite brushes by combining simple (and composite) brushes using logical AND or OR combinations.

As an important aspect of the system SimVis, the degree of interest (DOI) as defined by brushing is not restricted to binary values but may take any value within 0.0 and 1.0. Motivated by the idea of fuzzy logic [288], this concept is called *smooth brushing* [53] and is also supported by our approach. The user can split the border of a simple brush into independently modifiable interior and exterior boundaries, which causes the according DOI function to drop off linearly from 1.0 inside the interior boundary to 0.0 at the outer boundary. A visualization of the DOI in 2D scatterplots is essential: Brushes defined in the proposed

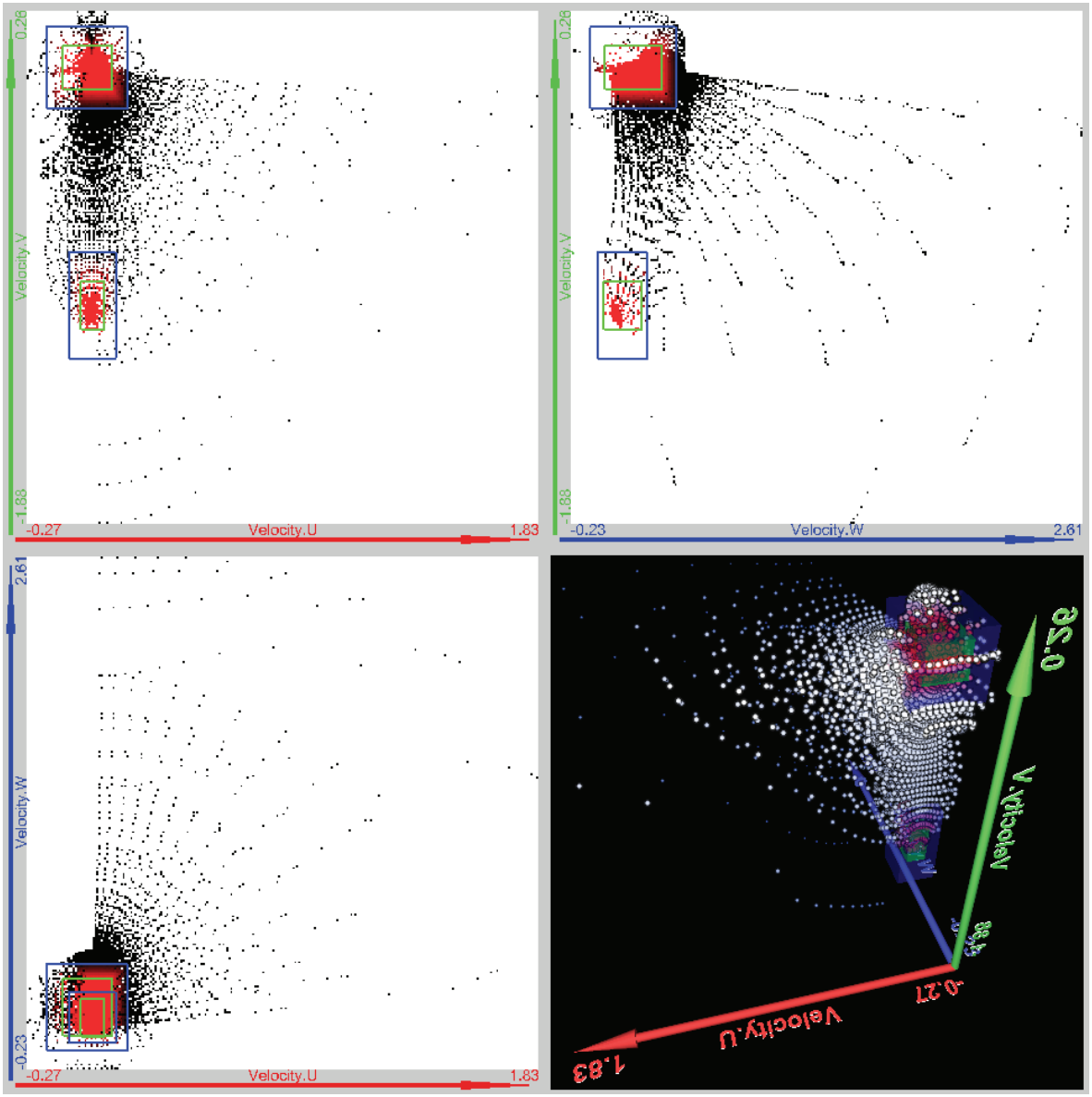


Figure 4.8: Composite smooth brushing in the combined 2D/3D View: Two cuboid-shaped basic brushes are logically OR-combined. DOI coloring is applied in all views. The basic brushes are drawn as rectangles in 2D and as transparent boxes in 3D. Note the arrangement of the views recognizable by the color-coding of the axes.

setup constrain three dimensions and thus one more than can be shown by a single 2D view. However, 2D rectangles as 2D outlines of brushes do not provide any information concerning the depth-validity of the brush. Even points inside a rectangle in 2D can lie outside the brush in 3D. Therefore the coloring of the points reflects their DOI in order to facilitate a correct understanding of the current brushing situation. Among possible visualization options for the DOI are mapping the minimum, maximum or average of the DOI values of all data entries, which project to a certain pixel, to color. Additionally, providing coloring based on the DOI

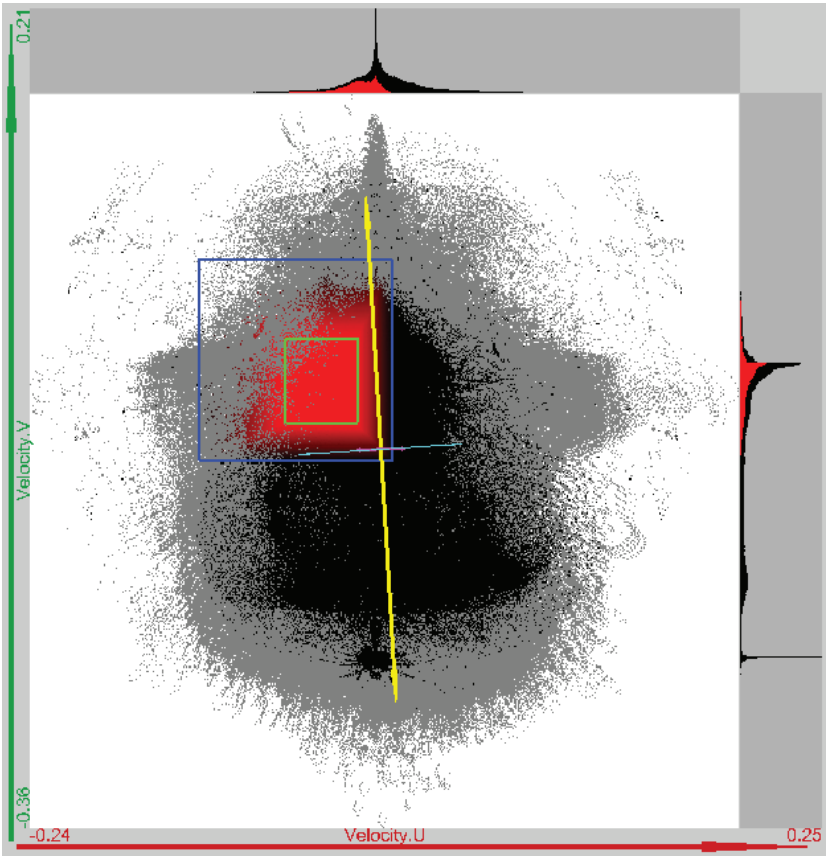


Figure 4.9: Extensions for 2D scatterplots: Histograms indicate the point density of the whole focus (black) and the brushed focus (red) and grey areas represent temporal context.

in the 3D view can further enhance the comprehension of the current brushing. An example for a composite smooth brush with DOI coloring in the combined 2D and 3D scatterplot is shown in Fig. 4.8.

4.2.2 Adapting 3D Extensions for 2D Scatterplots

Some extensions of 3D scatterplots as presented in section 4.1 are also applicable to 2D scatterplots in a slightly modified version. This is, because 2D and 3D scatterplots have some drawbacks in common. Due to the loss of one dimension compared to 3D scatterplots, the problem of unrecognizable point densities is usually even worse in 2D.

Analogically to the 3D case (see section 4.1.2), we depict the spatial focus with two histograms per view – for the X- and Y-axis, respectively (see Fig. 4.9). These histograms are located at the margin of each view and they share its resolution. A co-located and equally scaled histogram in a different color represents the distribution of the brushed subset of the data. Each data item is counted corresponding to its current DOI.

The principle component axes can also be shown in 2D views (see section 4.1.5). The user can choose between an independent 2D analysis for each view or an orthographic projection of the components as calculated in 3D into 2D. The former approach provides more information

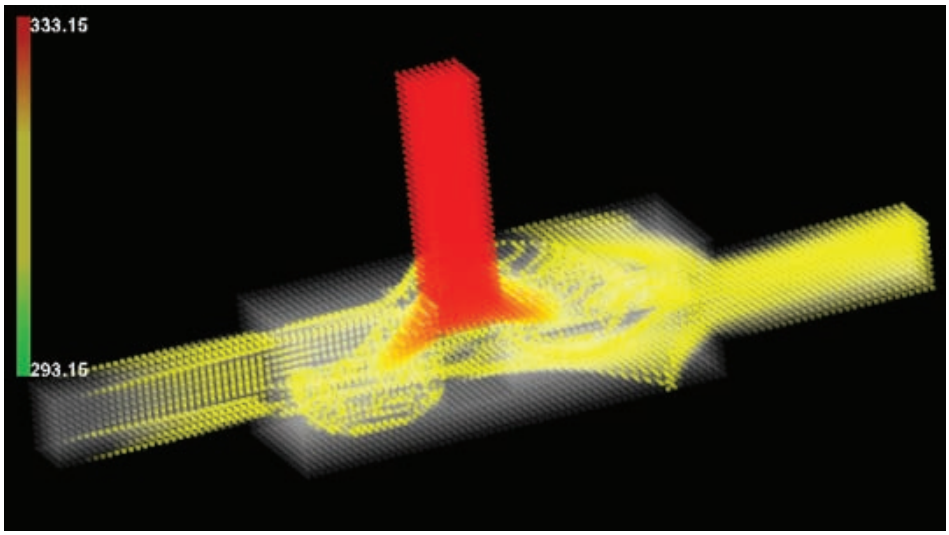


Figure 4.10: The T-junction: Warm liquid is entering from one inlet (right), hot liquid expands from a second inlet (above) and splits due to an obstacle.

for the 2D views themselves, while the latter is more consistent with 3D viewing.

Analogously to the 3D case, 2D scatterplots also permit visualizing the temporal context as described in section 4.1.4. Since the perception of the respective focus must not be compromised, we suggest drawing the focus on top of the context, hiding it in places where both focus and context can be found and using clearly discernable colors for both. Fig. 4.9 illustrates a combined application of the proposed 2D extensions.

4.2.3 Linking External Views

The combined 2D and 3D scatterplot is considered a single visualization by the SimVis system, even though it consists of four sub-views. The SimVis system employs a multi-level focus+context approach [186] and the coordination between these sub-views as described in section 4.2.1 is one of these levels. In addition, composite brushes can be defined by logically combining brushes of different views [53, 52]. We thus distinguish between local brushing (the combination of all brushes defined in a single view) and global brushing (actually brushed data points considering the overall composite brush). To allow for this distinction, different colors are assigned to locally and globally brushed data points (see Fig. 4.13 and Fig. 4.14 for examples).

4.3 Application Scenario

This section demonstrates the application of combined 2D and 3D scatterplots and other linked views for interactively exploring and analyzing a large dataset. It puts special emphasis on using the extensions proposed in this chapter as well as on the aspect of linking with other views of SimVis.

Our collaboration partner belongs to the field of the automotive engineering industry, where results from computational flow simulation are analyzed, which is a challenging task.

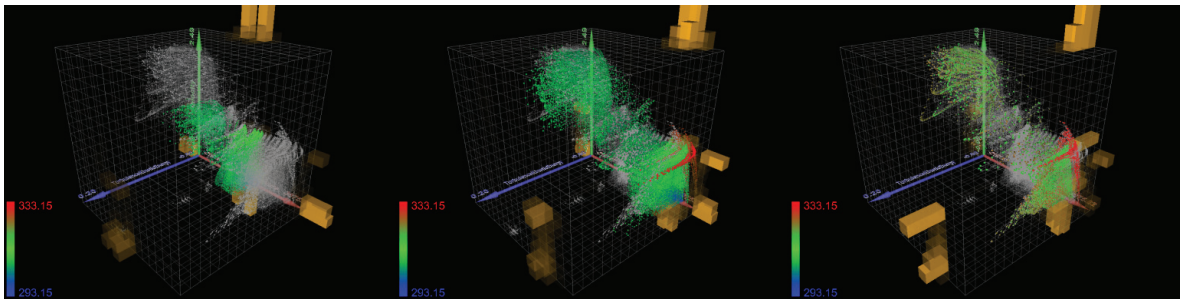


Figure 4.11: The development of velocity, pressure and turbulence over time: The first consolidation phase (left), the second expansion period (center) and the state at the end of the second consolidation phase (right).

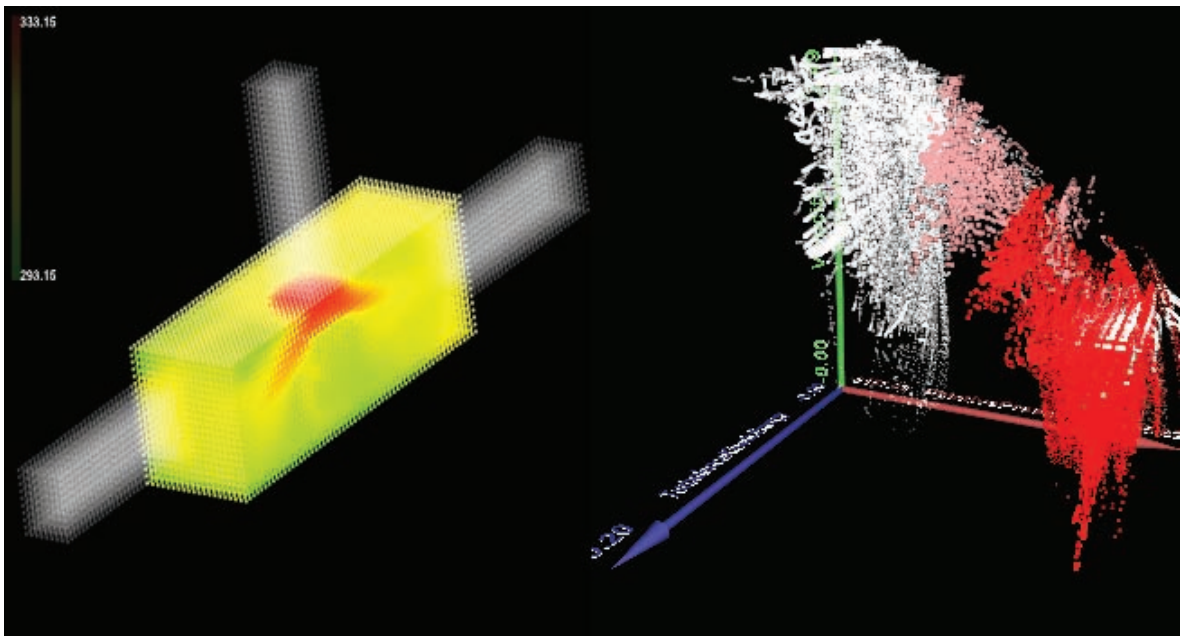


Figure 4.12: Mapping brushed physical space to attribute space: The main chamber of the T-junction (shown in the linked 3D View of SimVis) closely corresponds to the lobe at high pressure in the linked 3D scatterplot.

Simulations are time-consuming and typically many simulation-cycles are required to optimize the performance of a specific system. In order to speed up these simulation-cycles and in turn also shorten development times, interactive visualization is crucial to achieve fast and successful analysis of the data. Visualization often is the mean to understand complex relationships between different data items. However, the visualization has to cope with an amount of data which is usually vast due to detailed geometrical models (in terms of numbers of cells), the number of attributes computed for each cell and the time-dependent aspect (number of time-steps) of the simulation.

The case presented here is a T-junction with an extended chamber around the junction

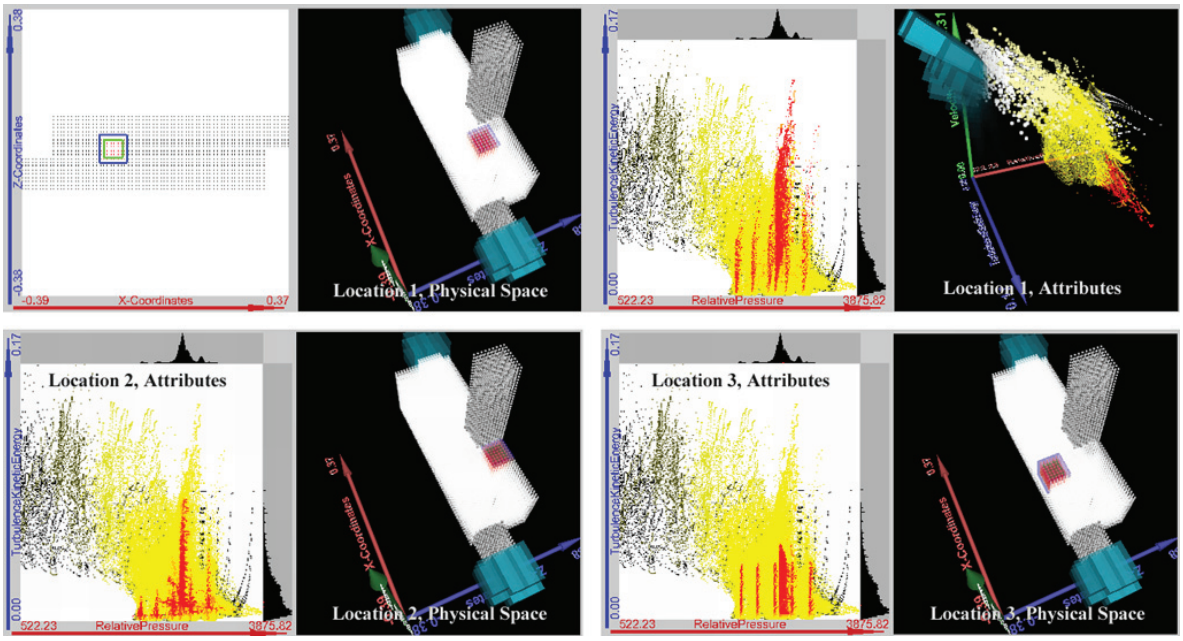


Figure 4.13: Relating locations inside the main chamber to their attributes: The turbulence kinetic energy (TKE, mapped to the blue axis) is very high for location 1 (top), considerable for location 2 (bottom, left) and only medium for location 3 (bottom right). Note the representation of spatial context in the 3D views. Not shown by this figure: The smaller lobes in TKE originate from time-steps before the hot liquid has fully entered the main chamber and vanish afterwards.

(see Fig. 4.10) and an obstacle below the secondary inlet (red in Fig. 4.10). Warm liquid starts to float in from the main inlet with the beginning of the simulation (at the right of Fig. 4.10). After the first third of the time-span of the simulation, a hot liquid enters the junction from the second inlet. In this example the user is interested in the mixing behavior and particularly in the existence of vortices and eddies, which are hard to detect by purely mathematical means. The dataset comprises 18 data attributes for approximately 32000 cells and 100 time-steps.

While the analyzed dataset comprises 18 dimensions, most questions can be answered considering a much smaller set of dimensions. A typical set for computational flow simulation involves velocity, pressure, turbulence kinetic energy (TKE), temperature and time. In our context, velocity, pressure, and TKE are mapped to the three axes of the 3D scatterplot, temperature is mapped to color and time is considered as temporal focus and context, defined with a range slider (see section 4.1.4).

In order to get a first approximate idea of the data, the exploration starts with moving a narrow temporal focus (time-slab) back and forth in time (temporal context and histograms for the spatial focus are enabled to permit a correct assessment of the relative position in time and of the current point distribution, respectively, see Fig. 4.11). This basically reveals four temporal periods: Expansion of the liquid from the first inlet towards the outlet, a first consolidation phase, expansion and mixture from the liquid of the second inlet, and the second consolidation phase. The two expansion phases are characterized by approximately linear

changes in velocity, quick rising in pressure and high values of TKE. During the consolidation phases, velocity and pressure remain approximately the same, while the TKE slowly decreases with increasing duration of the phases; the simulation shows stabilization at the end of the last phase. A particularly strong increase in the overall size of the point cloud with respect to the displayed dimensions can be observed during the second expansion phase, which suggests distinct vortices. This is identified as interesting for further examination – together with the beginning of the second consolidation phase in order to check, which turbulences are nonrecurring and which are persistent.

More advanced investigations require extended brushing facilities and linking of different views (see section 4.2.3). Relating physical locations to features in attribute space and vice versa is an essential part of the exploration with brushing. Three settings are reasonable in general for linked views:

- Brushing in space and visualizing the according attributes (which characteristics can we find in certain parts of the geometric model?)
- Brushing attributes and visualizing the related positions in space (where can we find certain characteristics?)
- Brushing attributes and visualizing other attributes (how are certain dimensions related to each other?)

We focus the further exploration on the main chamber (by brushing in space), since the situation within the inlets and outlet (being significantly influenced by the boundary conditions of the simulation) is rather known and therefore less interesting. Brushing in 3D space can easily be done with the combined scatterplot by mapping the X-, Y- and Z-coordinates of the cell centers to the three axes. As can be seen in Fig. 4.12, the main chamber (brushed in space and visualized with the linked 3D view of SimVis) largely coincides with the lobe at high pressure of the linked combined scatterplot showing pressure versus velocity versus TKE (Fig. 4.12 is an example for linking multiple views of different kinds). In order to make efficient use of the available space and resolution, we adapt the scaling of both combined scatterplots (space and attributes) so that they contain only the brushed part and consider the rest as spatial context.

Locally investigating the main chamber with a spatial brush which is refined to 5x5x5 cell centers (plus one surrounding slice of smoothly brushed cells) reveals high peaks of TKE in some parts next to the obstacle in the direction to the main inlet: Comparing spatial and attribute domain, Fig. 4.13 illustrates very different turbulence-conditions across the width of the T-junction over time. High turbulences often indicate the presence of eddies; therefore it could be that some of the hot water entering the main chamber from the secondary inlet forms an eddy in this place.

We try to verify or refute this assumption by brushing in attribute space: If an eddy exists, it most probably contains some of the entering hot liquid due to the proximity to the second inlet. We account for this consideration by brushing high temperatures in the histogram view of SimVis [153], as illustrated in the top of Fig. 4.14. Furthermore, we are especially interested in areas where the flow direction is different from the main direction, because swirling liquids usually exhibit velocities from a wide range of flow directions. In order to address this task, we refine the brush on temperature with a 3D brush defined in a linked combined scatterplot where the X-, Y- and Z-components of velocity are mapped to the three axes: The principle

component analysis of these three dimensions reveals that the main flow can be found in positive X-direction with an additional flow in negative Y-direction – it is also helpful for this task to display the zero crossings of each dimension (as thin lines in 2D and transparent planes in 3D). Smoothly brushing all opposite flows by selecting approximately the negative X- and the positive Y-direction as well as a certain band around zero in the Z-direction (Fig. 4.14 center) outlines a rotating flow, located in the investigated area, in the linked 3D view of SimVis (Fig. 4.14 bottom). This is a clear indication for an eddy, which cools down during the upward flow, as can be seen when mapping temperature to color. However, the visualization shows that the defined properties can be found on the other side of the obstacle as well: This secondary result could be subject of further investigations.

Concluding, we have demonstrated in short, how interactively linking the combined scatterplot with other views of SimVis has been a successful way to explore and analyze complex, multi-dimensional and time-dependent flow data. Apart from linking and brushing, a task-centered application of the proposed extensions to 2D and 3D scatterplots has proven useful in gaining an in-depth understanding of the dataset.

4.4 Discussion and Future Work

This chapter introduced several extensions to 2D and 3D scatterplots in order to alleviate various known restrictions. The introduced extensions improve the depth perception in 3D, address the problem of unrecognizable point densities in both 2D and 3D, and they help the user to keep an overview in space as well as in time when focusing on certain temporal and spatial parts of the plot. As an additional contribution, a tightly linked setup of 2D and 3D scatterplots intends to combine the advantages of both. The linked 2D scatterplots complement the 3D scatterplot by providing an alternative visualization as well as a convenient way for defining brushes which are consistent with the 3D setting. This chapter also described an integration of our approach in the SimVis system as an example of an interactive, multiple-view exploratory software. Finally, we demonstrated the application of the proposed scatterplot extensions as well as linking with external views by means of an exemplary investigation of large computational flow simulation results.

Although much less common than 2D scatterplots, 3D scatterplots have proven advantageous along with the presented extensions. The three dimensions match the dimensionality of physical space, which permits to intuitively visualize and interact with according data, like brushing certain areas of a physical model as well as the three components of velocity, for example, or any kind of three-dimensional gradients. The major drawbacks of 3D scatterplots are occlusion and difficulties with respect to comprehension and interaction due to the mismatch in dimensionality of three shown by the scatterplot and two used by standard input and output devices. This chapter showed an attempt to mitigate these problems.

Ideas for future work include additional 3D brushing concepts and an extension of the integration of the principle component analysis as a means to introduce a data-driven coordinate system. Being able to define principle-component aligned brushes could ease the specification of features.

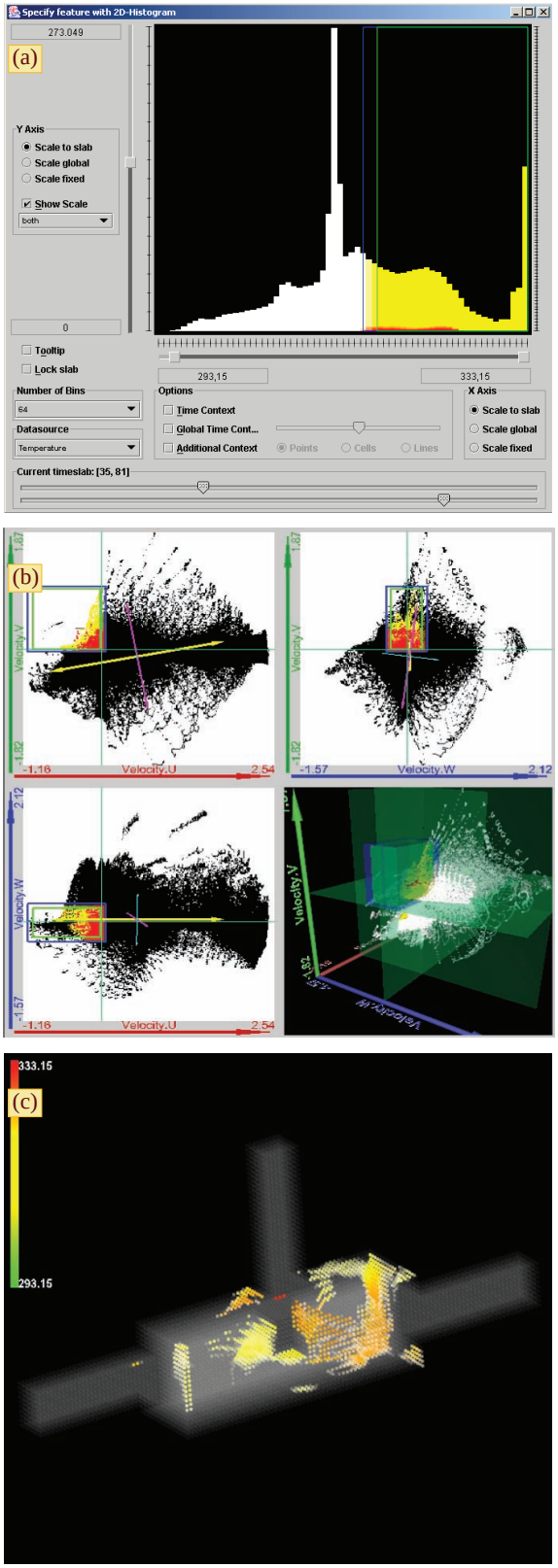


Figure 4.14: Combining several brushes to extract the eddy: Temperature is brushed using a histogram (top), the flow direction is defined with the combined scatterplot (center) and the result is visualized in the 3D view (bottom).

Chapter 5

Hierarchical Difference Scatterplots

Data dimensions of multivariate datasets can roughly be distinguished as being either continuous or categorical. While the data of some application fields is predominantly continuous (e.g., physical quantities), many application domains have to deal with mixed data, which has many categorical as well as continuous attributes (e.g., data from Customer Relationship Management). In this case, pivot tables are widely used to summarize the values of continuous attributes with respect to a classification given by categories. On-Line Analytical Processing (OLAP) [45] uses categorical attributes, called *Dimensions*, to split the data before aggregating continuous attributes, called *Numeric Facts*. An important aspect of OLAP systems is to use large-scale overview summaries of the data as starting point for selective drill down into interesting parts of the data.

OLAP is based on the fact that categorical data is closely related to hierarchical data and selective drill down (and roll up) is thus related to navigating a hierarchy. Apart from inherently hierarchical categories (e.g., years can be subdivided into months, days, hours, etc.), dimension composition is the key approach for defining hierarchies as it allows for specializing the categories of one attribute by the categories of another one. For example, two separate attributes "sex" and "age group" can be combined to obtain a category like "female and younger than 30". In the context of information drill down, pivot tables are also hierarchically structured and often referred to as data cubes (or OLAP cubes).

As described in chapter 2, a major benefit of pivotization is a lossless data reduction that enables to represent billions of underlying data items by simple visualizations. In general, however, this excellent visual scalability comes at the cost of a huge loss of detail. Interactive analysis tools for pivot tables should consequently support navigation in a way that it is up to the user to decide where to drill down and where to stay at a summary level. They should reflect this hierarchical aspect in the visualization.

Apart from the navigation within the hierarchy itself, a frequent analysis task is to compare categories within one hierarchy level and also between multiple hierarchy levels. The difference of pivoted values with respect to parent categories may characterize individual categories very well as demonstrated by common statements like "the average income in a particular region is x percent higher as compared to the entire country". A visualization approach for OLAP cubes should therefore also facilitate relating categories along the hierarchy.

Based on these considerations, this chapter introduces Hierarchical Difference Scatterplots (HDS) as a novel approach to the interactive visual analysis of OLAP cubes. The overall goal was to combine the visual scalability of overview summaries with the necessary degree of

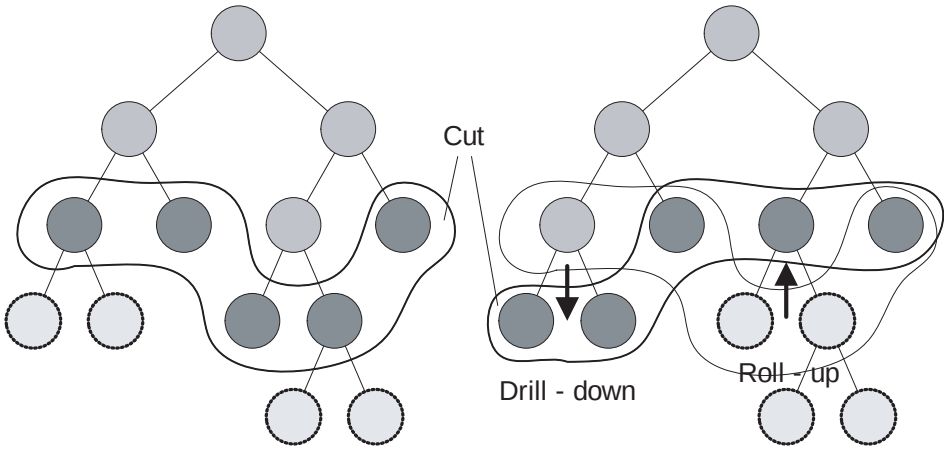


Figure 5.1: Navigating a hierarchy. Dark nodes represent the current state of navigation (the "cut"); nodes above the cut are contextual information and nodes below the cut are not visualized. Drill-down and roll-up operations transform the left hierarchy to the one on the right-hand side.

detail for selected parts of the data. The following list describes more concrete goals and tasks which guided the design of HDS:

- Relating categories to siblings and to parent categories with respect to two continuous attributes. Our consideration is that differences between pivoted values of parent and child categories provide an intuitive way of comparison. We therefore represent them explicitly.
- Integrating multiple hierarchy levels into a single visualization in order to analyze hierarchy levels in the context of the other levels.
- Supporting local drill-down and roll-up (see Fig. 5.1). Unlike other hierarchical visualizations, it is an essential aspect of HDS to provide different levels of detail for various parts of the data instead of representing the entire hierarchy as such. This is in accordance with drill-down tasks in huge OLAP cubes, which also often emphasize depth rather than breadth.
- Supporting a setup of multiple linked views in order to dynamically integrate results of arbitrary queries as defined by the user in linked visualizations (e.g., a certain cluster of customers of a sales dataset as selected in parallel coordinates).

Our clear focus is on supporting specific OLAP tasks by a combination of visualization and interaction. It is explicitly not the goal of HDS to be superior to existing tree visualizations with respect to providing visually pleasing still images of huge hierarchies as a whole. For tasks where this is required, we discuss, how other types of hierarchical visualizations can be tightly coupled to HDS. As one of many potential application scenarios, we evaluate our approach by analyzing a real-world social survey regarding national identity. The analysis has been conducted in collaboration with a social scientist. We also provide a discussion of analysis tasks as supported by HDS, limitations, and a motivation for visualizing differences explicitly.

5.1 Related Work

As described in section 2.2.1, *pivot tables* have long been used to summarize data. They have been extended to n-dimensional data cubes, which have widely been adopted by On-Line Analytical Processing (OLAP) [45]. Section 2.2.1 also described Polaris [238] and its commercial version Tableau as successful approaches of an interactive visual analysis of data cubes. However, Polaris displays a single level of detail (i.e., hierarchy level) and thus does not support comparisons between different levels of detail. The authors of Polaris also describe design patterns for adapting visualizations of data cubes on multiple scales [240]. This work deals with transitions between levels of details while still showing a single level of detail at a time. It has been mentioned as future work to communicate parent-child relationships and to deal with non-uniform branching factors.

The current version 6 of Tableau, however, does support comparisons between hierarchy levels using sub-totals and grand-totals, which are displayed in additional rows and columns. As the main drawback of this approach, comparisons require the user to look at multiple places on the screen in a successive manner. This makes comparisons difficult as will be discussed in more detail in section 5.5. This problem is inherent for approaches that rely on showing absolute values in a side-by-side manner. Therefore, visualizing differences explicitly was a main consideration in the design of HDS.

Yang et al. [285] propose a general framework for interactive hierarchical displays of large multivariate datasets, and they apply this framework to extend parallel coordinates, star glyphs, scatterplot matrices, and dimensional stacking. This approach categorizes a dataset by clustering before using this classification for multi-resolution analysis of aggregated values. However, unlike HDS as introduced in this chapter, Hierarchical Parallel Coordinates are limited to comparing results along one cut through the hierarchy, while our approach focuses on differences between levels. Sifer [228] proposes parallel trees, which employ a parallel axes layout for aligning multiple drill downs into a data cube. The categories of all hierarchy levels are stacked on top of each other. For analysis, the user may relate one active dimension to all others by coloring parts of the boxes. This implicitly conveys the information for comparing siblings as well as child categories to parent categories. Differences are not represented explicitly which requires remembering one category and shifting the attention to another one for comparison. This becomes even more difficult as categories are scaled in proportion to their relative frequencies and thus their size may differ significantly. Moreover, parallel trees require categorization of continuous dimensions (i.e., facts) and do not support typical aggregations like average or sum. This severely limits their applicability to frequent OLAP tasks.

Section 2.2.1 briefly summarized the huge amount of literature on the visualization of hierarchies and hierarchically structured data. While common node-link representations [12, 116] focus on showing the structure of the hierarchy rather than pivotized properties of nodes, containment-based approaches like tree maps [225] convey the size of the hierarchy nodes very well. However, only a few approaches derive the node placement from multivariate properties of the data as necessary for typical OLAP tasks.

Wattenberg proposes PivotGraph [270] for analyzing multivariate graphs and he addresses OLAP by supporting drill-down and roll-up. The graph layout corresponds to a grid which is given by two categorical dimensions for the X and the Y axes, respectively, and edge thickness is determined from the number of edges being aggregated. While the basic idea of property-based node placement is similar to HDS, there are several differences. PivotGraph only supports placement based on discrete dimensions while HDS uses a node layout scheme suited

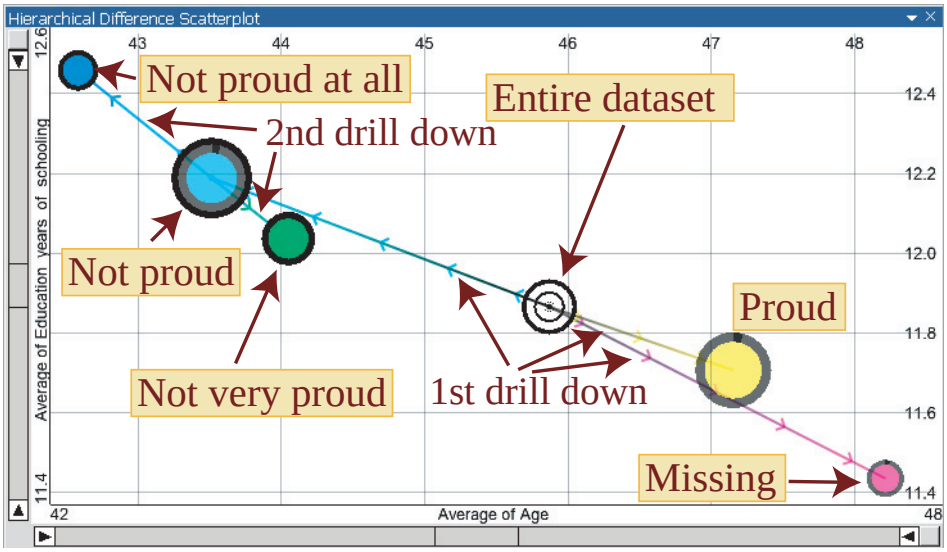


Figure 5.2: A simple hierarchy as conceptual example: the average age (X-axis) and the average years of schooling (Y-axis) are compared for several degrees of pride on armed forces and the entire data. Drill-down on "not proud" distinguishes "not very proud" and "not proud at all". The size of nodes shows the number of respective interviewees. The visualization reveals that pride is increasing with age and is decreasing with education.

for comparison of differences between continuous facts. Moreover, PivotGraph visualizes a single level of detail at a time (similar to Polaris [238]) and thus does not allow for relating nodes to their parents. After all, the intention of PivotGraph is to improve the interpretability of the graph topology for a particular level of detail, while HDS focus on comparing aggregated facts along and across a categorical hierarchy.

Queries defined through interaction within visual representations (also known as "brushing") are a proven standard approach for the identification of selected data subsets of interest (see section 2.4.1). However, there has been little research on integrating brushed subsets in hierarchical visualization techniques. In particular, no approach explicitly characterizes brushed subsets by displaying the difference between the properties of an entire category and its selected part.

5.2 Hierarchical Difference Scatterplots

This section introduces Hierarchical Difference Scatterplots (HDS) as a novel combination of scatterplots and tree visualizations. After describing the approach itself, we provide examples of tightly coupling HDS to other hierarchical visualizations and propose techniques for linking our technique to other multivariate visualizations.

5.2.1 Visualization

The main idea of HDS is to layout nodes of a tree based on properties similar to a scatterplot (see Fig. 5.2). For parameterization, HDS require a pre-defined hierarchy, i.e., a data cube, and several properties, which are assigned to the visual attributes X-position, Y-position, size,

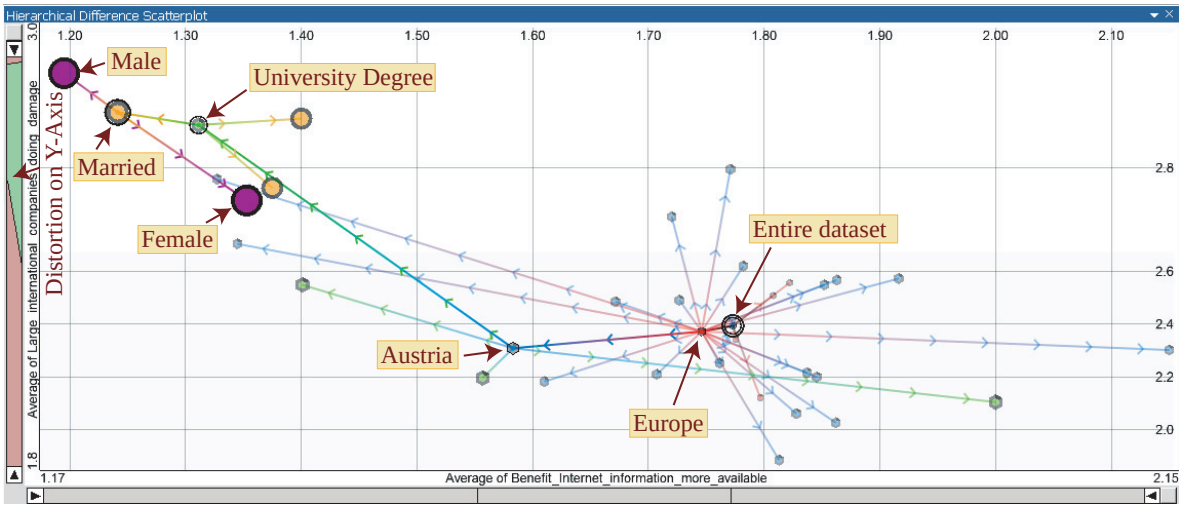


Figure 5.3: Example of a deep drill-down: the focus is on comparing men and women of the category path Europe – Austria – University degree – Married (i.e., five levels of the hierarchy plus the root) with respect to their attitude towards the Internet and international companies. Size, color and opacity are used to visually discriminate hierarchy levels. All siblings along the path are shown as valuable context information. Distortion is used on the Y-axis.

and color. Properties may be pivoted values of continuous data attributes. An example are aggregated "measures" like the average revenue per node or other aggregates like minimum, maximum, median, sum, etc. Other examples include inherent features of hierarchy nodes like absolute frequencies or depth. Applying data-driven glyph placement [267], the properties assigned to the X- and Y-attributes are directly mapped to the position of the visual representations of categories. In addition to X- and Y-position, the user may independently assign different properties to size and color which is comparable to Polaris [238], or use default settings. For example, size per default represents the number of raw data items for each node. Color is discussed further below.

In accordance with the idea of information drill-down, the user may increase the complexity incrementally and selectively. Initially, the entire data cube is handled as a single category and it is thus shown as one visual item. By clicking on this item, the user may drill down to the next hierarchy level that displays the respective hierarchy nodes as additional visual items. Clicking on any of these items adds its direct children and thus increases the amount of shown information locally for this particular sub-tree (see Fig. 5.2). As most important aspect of HDS, the visualization is not limited to the categories within the current state of navigation in the hierarchy (referred to as "cut", see Fig. 5.1), but also includes all nodes above the cut up to the root of the hierarchy. This allows for direct comparison of properties between child nodes and parent nodes as both are displayed in the same visualization and thus share the same visual context with respect to node placement. However, this necessitates concepts for discriminating levels of the hierarchy and recognizing structural relationships, which we address in multiple ways.

First and foremost, lines connect each parent to all visualized children, thus representing the topology of the hierarchy. In order to improve the distinction of lines in densely populated areas, connection lines smoothly blend the color of the parent to the color of the

child. As interesting aspect, these directed lines could be seen as "skeleton" of the visualization, which sketches the structure of the scatterplot of non-aggregated raw data entries. Even more important in the context of OLAP, the lines explicitly visualize the difference between the properties of each category with respect to its direct parent category (or the root of the hierarchy). Both the length and the angle have semantics, namely the overall amount of difference and the ratio. Due to the 2D layout, the lines support the perception of relationships between differences on the X and the Y axis. This allows for fast identification of sub-categories deviating in the same way from their parents for multiple sub-trees. In our implementation, optional small arrows pointing towards the respective child indicate the direction and facilitate tracing the structure of the hierarchy in some situations at the cost of increased clutter.

As mentioned above, each visual attribute can be used in different ways. In particular, each attribute can be used to enhance the discrimination of hierarchy levels, where transparency can be modulated independently from color. Transparency and size-based discrimination amount to a focus+context approach. One hierarchy level C is considered to be the current one, which is drawn opaque and in full size. Opacity and size decrease for lower and higher levels N with a factor of $1/2^{|C-N|}$. The current hierarchy level is a global property of the visualization, i.e., the same depth is highlighted through all sub-trees. Drill-down and roll-up operations automatically update the current level, or the user may manually set any level as current. Expanded nodes, i.e., nodes above the cut, are highlighted by an additional opaque circle. Directed lines leading towards expanded nodes are always drawn in full opacity (see Fig. 5.3), which facilitates tracing individual sub-trees as generated by a local drill-down.

HDS offer various modes for coloring hierarchy nodes. In addition to representing common categorical properties like size or pivoted values of an arbitrary measure as mentioned above, users may optionally also emphasize the structure of the hierarchy. Hierarchy-based coloring recursively subdivides the hue circle in a similar way as described for the Interring [284]. The segment of the hue circle assigned to each node is proportional to the number of leaf-nodes in the sub-tree and the hue in the middle of the segment is applied to the node itself. Color is a particularly important issue when coupling different tree-visualizations, as it supports the visual matching of hierarchy nodes (see section 5.2.2).

With an increasing number of displayed nodes, the extents and the density of the visualization may vary significantly during the analysis. Restricting the displayed value range in a similar manner as in Spotfire [1] is supported by our approach, but it has the disadvantage that users may lose the overview because the entire hierarchy is not visible any more. As an alternative, we offer spatial distortion in a similar way as Table Lens [207]. This has proven useful to provide focus+context for areas where nodes with similar properties are close together. Applying a piecewise linear visual transfer function [34], the user may smoothly magnify any contiguous sub-interval of the displayed value range. The factor is chosen separately for the X- and Y-axis (see Fig. 5.3 and 5.4). The reason for using a piecewise linear function instead of using non-linear distortion (e.g., fish-eye distortion [161]) is that differences between nodes remain comparable as long as all involved nodes are inside the focus, which can easily be ensured by the user.

5.2.2 Coupling Tree Visualizations

Arguably, no single visualization approach perfectly covers all aspects of hierarchical data. The clear focus of HDS is on supporting the interactive analysis of data cubes in the context

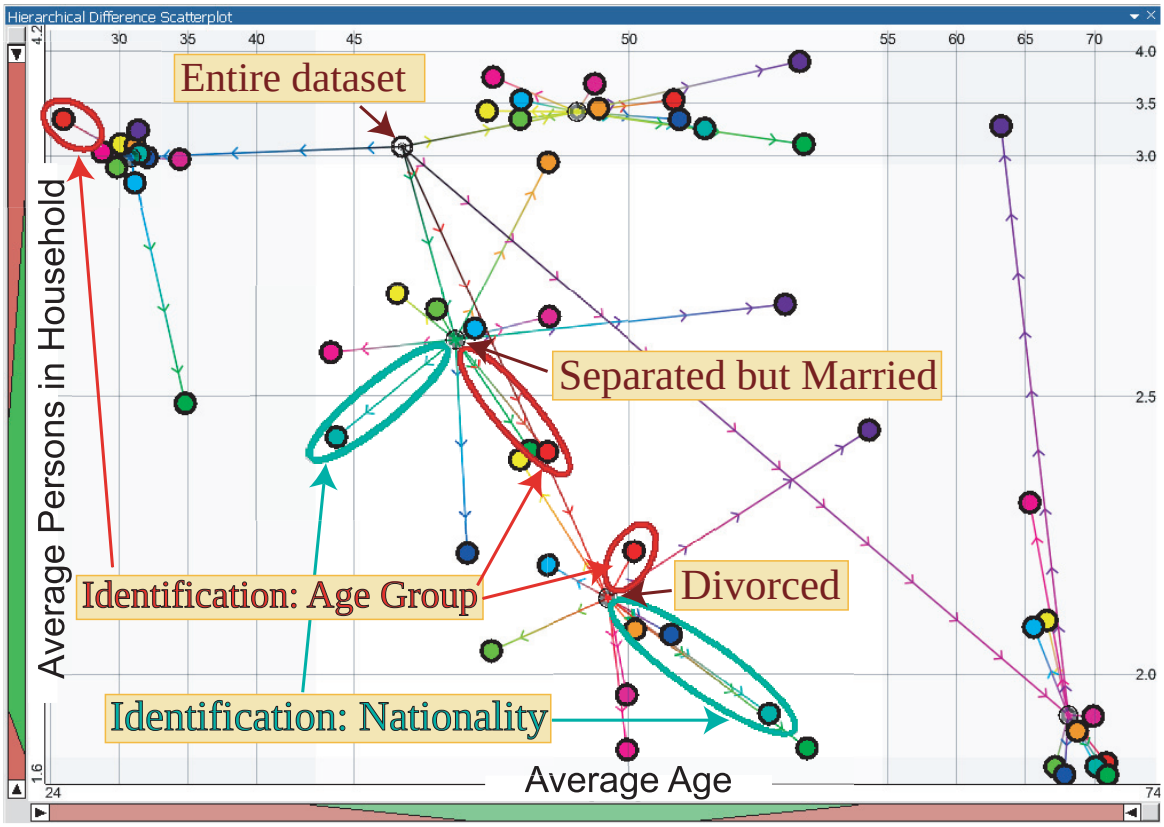


Figure 5.4: Comparing multiple sub-trees: interviewees are distinguished by their marital status and most important identification (in this order). Each class is characterized by its average age and the average number of persons in the household. While most identification nodes deviate roughly in the same direction for all marital status nodes, some interesting exceptions, like "Nationality", show contrary behavior for different nodes. Color is derived from the category name. Spatial distortion is applied on both axes to focus on divorced and separated but married interviewees.

of OLAP. By displaying multiple pivoted values (or other properties) and the differences to parent levels at the same time, HDS visualize comparatively much information per node. Due to the data-centric layout, however, HDS do not perfectly scale to the visualization of both depth and breadth of large hierarchies at the same time (i.e., the hierarchy as a whole). This is due to well-known graph-drawing problems like a potentially high number of crossing edges. However, as discussed in section 5.5, this is not a limitation with respect to analyzing large real-world data cubes, because the user may increase the complexity incrementally and selectively by drilling down to interesting details while staying at a coarse level for less interesting sub-trees (or even hiding them).

Still, aspects conveyed not so well by HDS might be interesting. We therefore briefly discuss concepts of tightly coupling HDS to other approaches for visualizing hierarchies in order to combine their benefits when analyzing the same hierarchy. As an example, we have implemented a layout similar to parallel trees [228] as used by Sifer to analyze OLAP data. This layout is related to ArcTrees [191], which we refer to as hierarchical bargrams since we

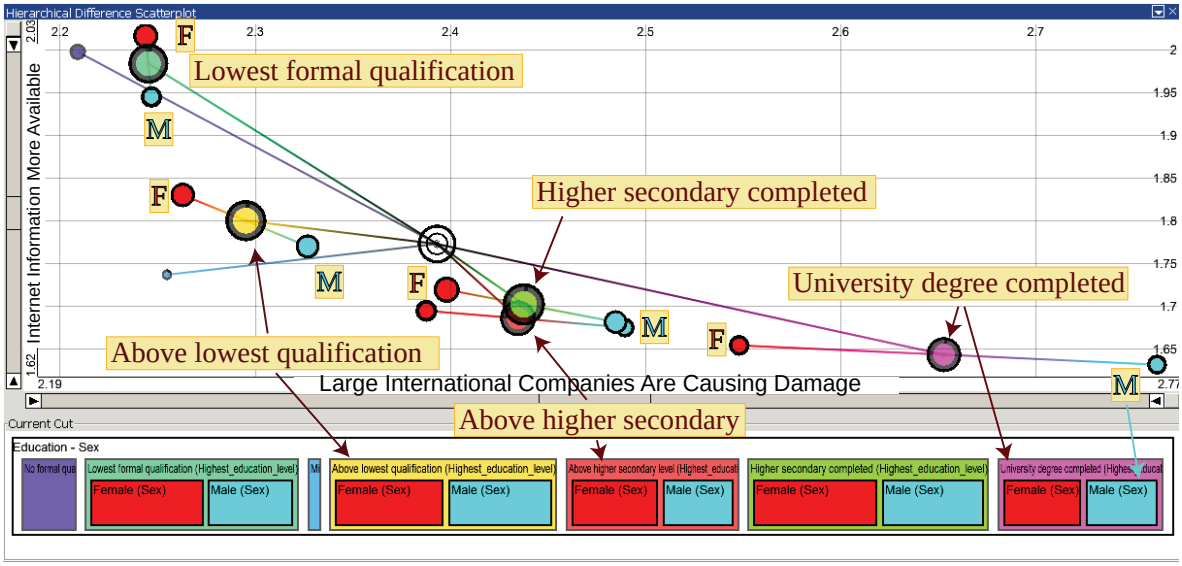


Figure 5.5: Tightly coupling HDS to hierarchical bargrams for displaying frequencies and names of hierarchy nodes. Several education levels, partly split into male (blue and letter "M") and female (red and letter "F"), are compared with respect to the average attitude towards international companies (X-axis, hue) and benefits of the Internet (Y-axis, saturation). The nonlinear relationship between the questions and the influence of education and sex are clearly visible.

do not show any arcs. In hierarchical bargrams, a horizontal bar representing 100% of the displayed data is subdivided in proportion to the relative frequencies of the categories in the first level of the hierarchy. The obtained boxes are recursively split in proportion to the relative frequencies of their sub-categories. This generates bars nested inside the representation of their parent-category. Each bar displays the name of the respective node (see Fig. 5.5).

We have identified the following attributes for tightly coupling HDS to other kinds of tree visualizations.

- **State of navigation** The user may perform drill-down and roll-up operations in any visualization, which consistently updates all views. In the hierarchical bargrams, the recursion stops at the current cut, which is also conceivable for most other types of tree visualizations (like tree maps).

- **Color** As discussed above, HDS offer multiple ways for using color. Applying consistent coloring of nodes to all visualizations greatly facilitates the visual matching between them. In our case, the bars in the bargrams are drawn in the same color as the nodes in the coupled HDS. Deriving the color from the position of nodes in the HDS (e.g., by mapping the position on the X-axis or the difference from the root to color) enhances the matching even more. Coupling by color is possible for almost all types of tree visualizations.

- **Order** Many tree visualizations have a degree of freedom in which order siblings are represented. This freedom can be used to roughly maintain proximities between nodes throughout all visualizations. The hierarchical bargrams, for example, optionally order

sibling nodes with respect to their position on the X- or Y-axis in the coupled HDS.

- **Selection** Interaction is generally very powerful for linking visualizations. We provide different types of selection: (1) based on dedicated mark up interactions (e.g., by drawing a rubber band or actively clicking on an item) (2) temporarily hovering over visual items, which highlights the node or sub-tree beneath the mouse cursor throughout all visualizations. This has turned out to be very intuitive and fast for matching nodes as no mouse clicks are needed.

5.2.3 Integrating Selected Subsets

The previous section discussed tightly coupling HDS to other tree visualizations. This section describes the integration of subsets as defined by brushing arbitrary multivariate visualizations like parallel coordinates. It also applies to linking multiple instances of HDS visualizing different hierarchies. Linking views by interactive queries has established itself as important concept, because different sub-tasks of a complex analysis typically require different types of visualization. For example, the user may want to identify multi-dimensional clusters in parallel coordinates, and immediately relate each cluster to a hierarchy as visualized by HDS.

In a linked setup, each type of visualization typically highlights the subset of selected entries in an appropriate way. In HDS, the integration is based on the fact that the selection state is categorical too. Each row in the underlying non-aggregated main data table is either selected or not at any point in time with respect to a particular query. Employing the concept of dimension composition, a selection thus refines any hierarchy node X into "X and selected" and "X and not selected". This allows for visualizing selections similar to normal child nodes.

For each node X of the cut, the aggregations of the selected part of X (unless empty) are computed and visualized at the respective position in the plot (see Fig. 5.6). As for actual sub-categories, a line connecting the representations of the entire category X and its selected part explicitly represents the difference between both with respect to pivoted values. In order to discriminate multiple selections, the border of selection nodes is drawn in the color of the respective query, while this part is black for actual nodes of the hierarchy. Immediately updating the visualization at each modification of the selection implicitly generates an animation of change similar to moving the time slider of the Gapminder Trendalyzer [89]. In our case it concerns general variation on arbitrary data dimensions. The modification speed of each node representation reflects the gradient of change with respect to the selection criterion. As recently discussed by Robertson et al. [213], it also reveals overall trends, e.g., all selection nodes move from left to right, and makes outliers discernable, which move in a contrary direction.

5.3 Implementation and User Interface

HDS have been implemented in the context of Visplore, an application framework for visually supported knowledge discovery in large and high-dimensional datasets. Visplore supports the analysis of datasets with millions of entries and hundreds of dimensions at interactive rates on consumer hardware. This has a major impact on the design of all views (including HDS) and necessitates advanced software techniques like multithreading (see chapter 3). Visplore also supports missing values and requires all views to do so.

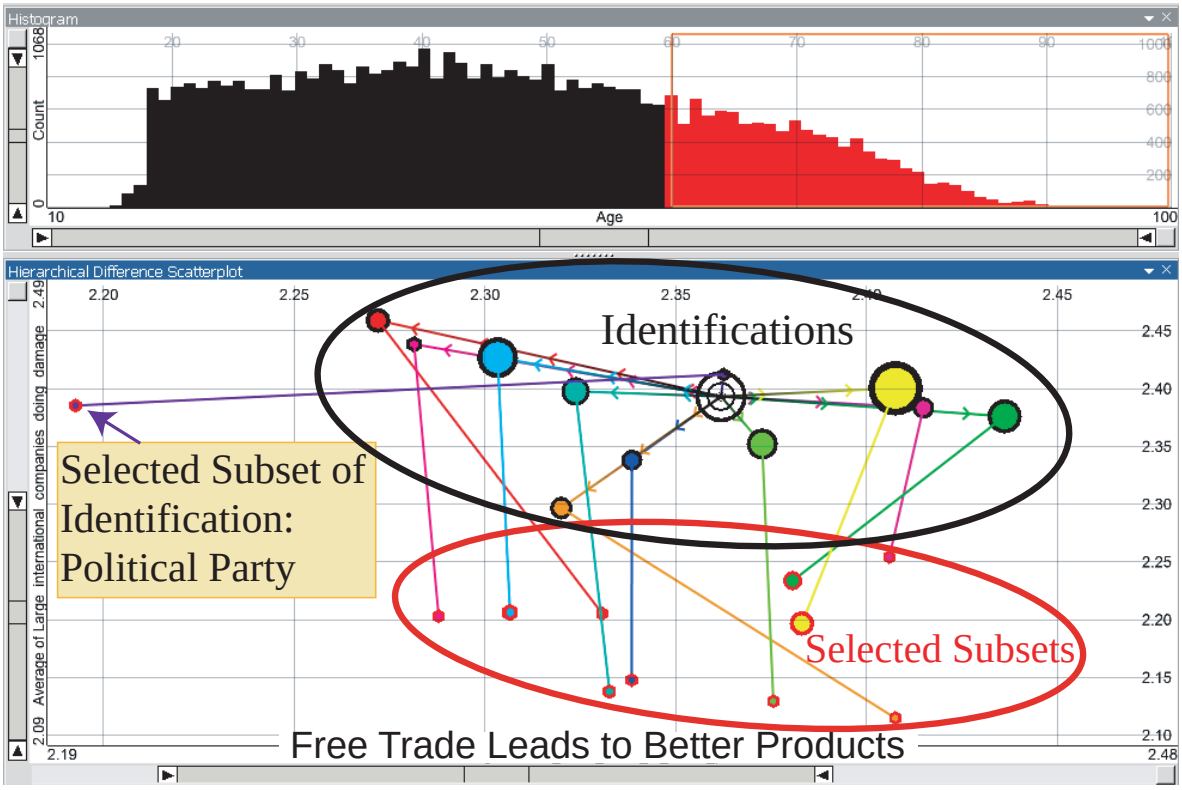


Figure 5.6: Integrating queries: interviewees older than 60 years, as brushed in a histogram, are highlighted for each category regarding most important identification with the average attitude towards free trade (X) and damage done by international companies (Y) assigned to the axes of the HDS. The visualization shows that elderly people tend to have an over-proportionally negative attitude towards international companies, while the attitude towards free trade is in most cases independent of age. The category "political party" is an exception, though, as the acceptance of free trade is higher for elderly people and – unlike for the other categories – more significant than the difference regarding international companies.

Visplore currently provides more than 10 different visualizations, which are partly standard (e.g., 2D and 3D scatterplots, parallel coordinates, histograms, etc.) and partly specific to certain application tasks (e.g., validating regression models as described in chapter 7). A key aspect of Visplore is to discriminate multiple queries, which are defined by composite brushing and are highlighted by all views in a linked way. All components also offer convenience functionality like undo/redo and a consistent way to arrange controls like data dimensions of the current dataset. In particular, the user may at any time specify new hierarchies of arbitrary complexity by dimension composition or by combining categories. Data dimensions and hierarchies can easily be assigned to views, which is the way how the axes and the displayed hierarchy of HDS are parameterized.

Making the user interface easy-to-use was also an essential design aspect of HDS. The user may perform drill-down and roll-up operations by just clicking on a visual representation, or may hide entire sub-trees. Tool tips provide details-on-demand showing the name, the size, and the aggregated values for the node beneath the mouse cursor. In order to highlight

subsets of the data in linked views, the user can brush nodes by either clicking on them or dragging a rubber band. Dedicated widgets next to the X- and Y-axis offer all functionality related to adapting the displayed value range and the spatial distortion.

5.4 Case Study and Evaluation

We now discuss the evaluation of our approach by the interactive visual analysis of a large survey, which we did together with a sociologist. The analysis of opinion polls is an important topic, where too little attention has been devoted to. HDS are designed to be generally applicable to data cubes of any kind, e.g., business data as a typical application of OLAP, and are not limited to opinion poll data. The sociologist had rich experience with the analysis of surveys, but had used static statistical software and had never used interactive visualizations before.

The survey was conducted by the International Social Survey Programme (ISSP) [206] in 33 countries between February 2003 and January 2005 with 44.170 respondents in total. Disregarding country-specific and thus incomparable questions, the dataset consists of 104 predominantly categorical attributes. The attributes are partly demographic questions and partly concern the attitude towards national consciousness, identity, and pride. The answers to most questions comprise 4 or 5 levels, e.g., very proud, somewhat proud, not very proud, not proud at all. This allows for both treating them as categories as well as computing meaningful aggregations, e.g., the average accordance to a statement. The dataset contains missing values, which represent an own category for categorical attributes. Missing values are disregarded when aggregating a continuous attribute.

Before analyzing the questions regarding attitude and pride, the sociologist first wanted to gain an overview about characteristics of various demographic categories, figures of the survey, and potential relationships between them. HDS facilitate this task, as it is fast to visualize simple pivot tables like the average number of persons in a household per country and they also quickly provide the size of each category. Within a few minutes, the expert could look at dozens of combinations, partly confirming expected facts, e.g., the average age of widowed people is 22 years higher than the average of the dataset. Partly, this basic analysis already revealed unexpected features like a significant variance in the average age of interviewees throughout the countries (which must be taken into account for subsequent conclusions).

Already for such flat pivot tables, the sociologist appreciated being shown the average of the entire dataset as visual reference. The reason is that this reference is not affected by categories of different size - a common problem when trying to determine the center in a purely visual manner (e.g., by assuming the center of the image as center of the data, which is typically misleading). As criticism regarding our implementation, the expert said that he lacked labels next to the nodes, although he admitted that tooltips partly compensate for that. We suggested using coupled bargrams as legend and deriving the order of the nodes from the X-position in the HDS. He made use of them for cases where only a few nodes are simultaneously shown, while they turned out to be of limited scalability for more complex hierarchies.

After analyzing cross tabulations between categories (a frequent task in sociology) in another visualization of our framework, the expert returned to HDS in order to characterize categories in the context of other categories. For example, he was interested whether differ-

ent categories concerning identification have a similar distribution of age for different marital status categories (see Fig. 5.4). Showing multiple hierarchy levels simultaneously and explicitly representing the difference between them turned out to significantly help answering this and other comparatively complex questions. Within a short time, the sociologist identified multiple interesting and unexpected facts in the data. Comparing the difference vectors of the red dots in Figure 5.4, for example, reveals that for all categories related to marital status, the subset specifying "age group" as most important identification tends to be older than the average of the entire category. However, singles are a remarkable exception, as the "age group" sub-category of singles is the youngest of all. Assigning the same color to related sub-categories (e.g., red to all "age group" sub-categories) greatly facilitates such comparisons between different sub-trees. As the visualizations became more complex, the sociologist used distortion increasingly often and found it a convenient way to clarify relationships for densely populated areas.

As the next step of the analysis, the expert was interested in results concerning attitude and pride. Figure 5.5, for example, shows that people with a positive attitude towards the Internet turned out to be less skeptical towards large international companies. It further reveals a strong influence of the education level. For drill-down scenarios involving more hierarchy levels, the sociologist liked that he could focus on particular categories but still see the rest as context information, as illustrated by figure 5.3. While focusing on Austrian interviewees with a university degree, still all other education levels are shown for Austria, all other European countries, and all continents. The center of the entire dataset is given as well. The expert considered such deep local drill-downs a key advantage of HDS. Analyzing the difference between two levels is of course also possible by visualizing this derived information in simple scatterplots. Relating four or five levels at a time, however, would generate numerous derived data dimensions, which are hard to analyze intuitively without HDS.

The sociologist needed some time to familiarize with the idea of specifying ad-hoc categories by brushing linked visualizations. He eventually embraced this approach and used queries as defined in linked visualizations frequently for two types of tasks:

- **Motion** Due to the immediate update, changing the query in one view generates an animation in HDS. Figure 5.6 shows an example, where interviewees are selected by age in a histogram. Moving the interval from young towards old makes the selected parts of most identification classes in the HDS wander from top to bottom, indicating more skepticism towards international companies for elderly people. It also reveals interesting contrary trends for "political party" and "ethnic background" regarding the attitude towards free trade in dependence of age.

- **Highlighting** When comparing multiple sub-trees, a convenient way of identifying related categories throughout all shown extracted branches is by brushing this particular category in a linked view. For example, instead of assigning the same color to related sub-categories in figure 5.4, it would also be possible to highlight all "age group" nodes by selecting the category "age group" in another view, e.g., in a second instance of HDS. Using the "Superfocus", the sociologist could identify many different categories in a short time. This was particularly useful when color was needed otherwise - for example to discriminate hierarchy levels as in figure 5.3.

Although we can only describe a small part of our analysis here, this application has demonstrated how HDS facilitate and speed up the interactive analysis of data cubes. As result of

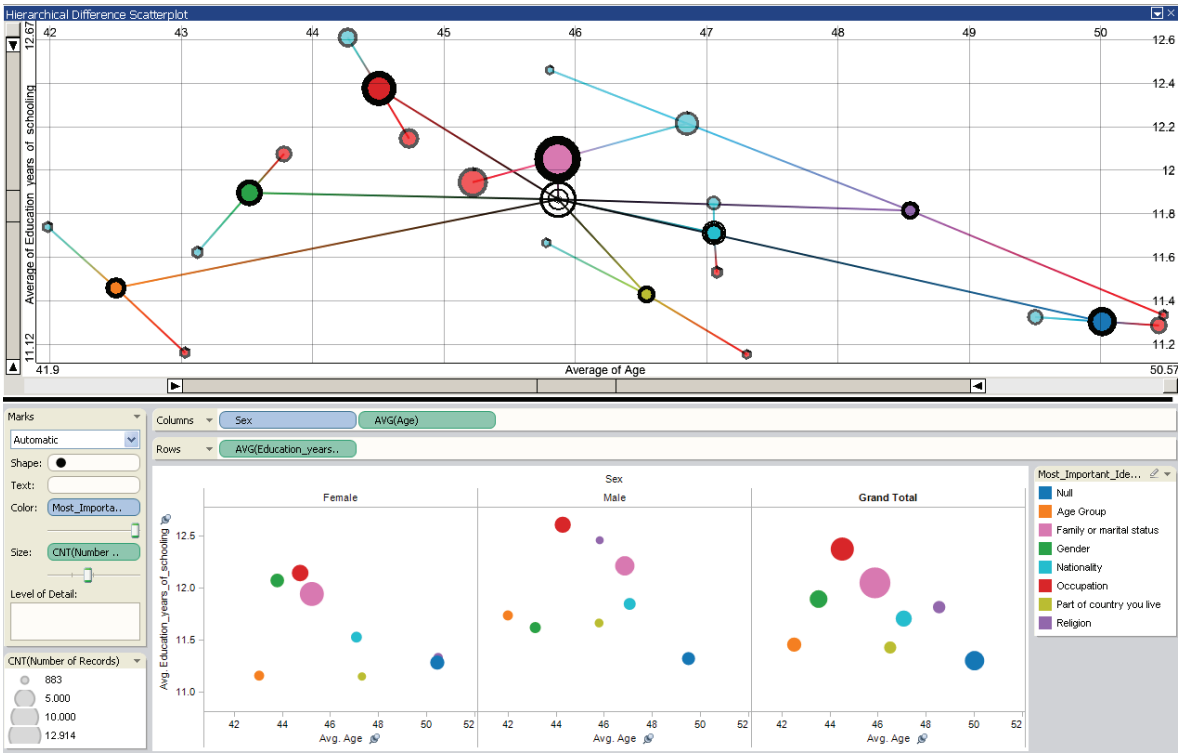


Figure 5.7: Comparing hierarchy levels using HDS (upper half) and using multiple scatterplots in Tableau (lower half). The average age (X axis) and the average number of education years (Y axis) are shown for groups having different most important identifications (color), which are further subdivided by sex. The same colors are used for corresponding identifications in both halves. In HDS, "male" is drawn blue and "female" red, while multiple panes are used below. Comparing especially the horizontal position of items is difficult across columns, while even minor differences are clearly conveyed by HDS.

our evaluation, the sociologist particularly liked being shown the center of the data as reference and being able to analyze multiple levels of the hierarchy in the context of each other. Despite tooltips and coupled hierarchy visualizations, his most important criticism concerned the lack of labels, which we will address in future work.

5.5 Discussion and Future Work

The main goal of HDS is to support the interactive visual analysis of data cubes. Selective drill down ensures that users can increase the amount of detail incrementally for sub-trees of interest. This is an important aspect regarding the scalability of HDS as it combines the visual scalability of overview summaries with the necessary degree of detail for selected parts of the data. As for all approaches relying on pivot tables, the speed for aggregating data is the most significant limitation with respect to the number of underlying data rows. Aggregating data is generally fast even for millions of data rows and may even make use of explicit optimizations for data cubes in data warehouses. Therefore, HDS scale well for datasets consisting of multiple millions (and even billions) of underlying data records, which

makes them applicable to real world data cubes.

A relevant question concerns the amount of detail (i.e., how many hierarchy levels and how many nodes), that can be shown before the visualization suffers from cluttering. An answer depends on the purpose. Generally speaking, HDS are suitable for:

- Comparisons *along the hierarchy*. The main intention is to relate a few particular nodes to their direct and indirect parent nodes. Such comparisons involve local drill downs of numerous hierarchy levels while typically little information is shown per level (see Fig. 5.3 for an example). In this case, the most interesting information is the path to the root node (i.e., the properties of the entire data cube). Siblings provide rather context information and it is often even tolerable to hide siblings for certain hierarchy levels. In this case, comparing ten or more hierarchy levels is possible. However, as shown by Fig. 5.3, the number of visual attributes needed for discriminating hierarchy levels generally increases with the number of hierarchy levels. As a special case, mapping the depth of a node within the hierarchy to its position on one of the two axes yields a common rooted tree layout. This layout is guaranteed to have no crossing edges as long as not more than one node is expanded per hierarchy-level.

- Comparisons *across one hierarchy level*. The focus is on the position of siblings relative to each other and to common parent nodes (see Fig. 5.4 for an example). Much information is shown for a single hierarchy level while little information – if any – is typically shown for other levels. In this case, HDS resemble non-hierarchical scatterplots, but may still convey additional information (e.g., the properties of the entire data cube as one additional item). In this case, comparing a few hundred categories is possible.

As a consequence of displaying much information per node (i.e., two pivoted properties and topology), HDS are limited with respect to showing both depth and breadth of large hierarchies simultaneously. Showing large hierarchies in their entirety was not a design goal of HDS and it is not necessary for many tasks. As discussed in section 5.1, most tree visualizations convey the topology but disregard multivariate attributes. Most approaches for OLAP, on the other hand, consider multiple attributes but are limited to displaying a single hierarchy level.

Tableau optionally displays multiple hierarchy levels using sub totals and grand totals which are added as additional rows or columns. However, comparisons require looking at multiple places on the screen in a successive manner. Generally speaking, comparisons become increasingly difficult and less precise with increasing visual distance and number of visualizations involved in the comparison. For example, while detecting even minor differences in the height of two adjacent bars of a bar chart is easily possible, comparing the position of points of multiple non-adjacent scatterplot panes is difficult and coarse. The reason is that the user is forced to "remember" one pane while shifting his focus to another – potentially distant – pane. Fig. 5.7 illustrates this aspect. Although three panes (as shown in the lower half) is quite a small number, precise comparisons are particularly difficult with respect to the position on the X-axis. The figure also shows that using a single row makes comparisons with respect to height much easier, because the same vertical reference is given for all items. Using multiple rows (e.g., by assigning identification to rows instead of using color) would severely compromise comparability of the Y-position as well. In the worst case, comparing panes might even involve scrolling the entire visualization. This problem is inherent for approaches that do not explicitly visualize the difference between items but rely on showing

multiple visualizations in a side-by-side manner as small-multiple visualizations typically do. Drawing items in a single scatterplot does explicitly visualize the difference between them, as this difference is directly proportional to their distance. This was a main consideration in the design of HDS.

There are multiple interesting directions for future work. First, we intend to conduct a large-scale user study in order to evaluate HDS more formally. Second, the issue of labelling nodes as mentioned by the sociologist needs to be addressed. The challenge is to add labels in a scalable way without compromising readability. Third, we plan to examine the effect of varying the shape of node representations on the interpretability of the visualization.

Other interesting questions for future research concern the applicability of HDS within small-multiple displays. A scatterplot matrix, for example, would allow for visualizing more than two measures at a time. Moreover, comparing drill-downs for multiple sub-trees of one node in a side-by-side manner could be an option for analyzing many deep drill-downs simultaneously. However, care will have to be taken as the aforementioned disadvantages of small-multiple visualizations apply in this case.

5.6 Conclusion

The analysis of data cubes is a key issue in many application domains. It involves navigating a potentially large hierarchy as well as comparing nodes within one or between multiple hierarchy levels with respect to properties like size and pivoted values. Particularly the difference between hierarchy levels is important information, which is not adequately represented by existing visualization techniques. Therefore, this chapter introduced Hierarchical Difference Scatterplots (HDS) as an interactive approach to analyze multiple hierarchy levels in the context of each other and to emphasize differences between them. Visualizing both the topology and two pivoted values per node, HDS display much information at a time. For many tasks, this means an added value as compared to alternative approaches. For example, analyzing differences between hierarchy levels using non-hierarchical scatterplots requires the user to look at multiple views (i.e., positions of the screen) in a successive manner. HDS display the difference between categories explicitly within one visualization, which makes comparisons more intuitive and more precise.

A key idea of HDS is to allow for incrementally and selectively increasing the amount of detail using local drill-down. This combines the visual scalability of overview summaries with the necessary degree of detail for selected parts of the data, ensuring that the proposed concept of HDS is reasonably applicable to data cubes of any size. HDS employ several focus+context approaches involving transparency, size, and distortion in order to ensure interpretability also for a significant number of displayed nodes. As other tree-visualizations are superior with respect to providing a pleasant layout of the entire topology or showing frequencies, we discussed concepts of tightly coupling HDS to other tree visualizations. Moreover, we discussed linking arbitrary other visualizations by user-defined queries to HDS. This allows for analyzing properties of ad hoc categories, it reveals trends through animations when changing queries, and it may also be used to highlight particular nodes. We described an evaluation of our approach by analyzing a large survey, which revealed numerous interesting and non-trivial aspects within a short time.

Chapter 6

Quantifying and Comparing Features in High-Dimensional Datasets

Various technologies address the highly non-trivial issue of extracting useful information from potentially huge datasets in different ways. Statistics have been used for long in order to provide summarized data characteristics. Basic statistical moments like mean, variance or correlation are very common and can be computed extremely fast even for millions of values on today's computers. However, statistics as such – and also most statistics-based techniques in the fields of machine learning – are quite static approaches as they hardly involve the user and typically yield a result without additional context information.

Information visualization follows a user-centric approach which is particularly suitable for exploratory analysis (see chapter 1). However, visual results are often only qualitative and thus not fully sufficient for many tasks. Additionally, to deterministically parameterize visualizations is a challenge itself when it comes to exploring high-dimensional datasets. Selecting a small number of dimensions to be displayed in low-dimensional projections like scatterplots may become a difficult task without a-priori knowledge or dedicated support.

The Rank-by-Feature framework by Seo and Shneiderman [222] is an example of a successful combination of statistics and information visualization. It addresses the issue of conveying a quick overview about all dimensions at an early stage of the analysis. However, its limitation to global features – statistical measurements are only computed with respect to the entire dataset – clearly restricts its continued application at later stages of the analysis, e.g., after identifying clusters or separating trends from outliers, where the user might be interested in properties of selected data subsets.

The main contribution of this chapter is an approach for characterizing and comparing arbitrary subsets by combining the precise information of well-known statistical moments with the expressiveness of visualization. Pursuing the concept of Seo and Shneiderman [222, 223], the statistics can be used for ranking small preview visualizations. Color enables a quick assessment of differences between dimensions or subsets and numerical values precisely characterize the respective data. We propose a 1D approach for univariate moments of individual dimensions and a 2D approach for bivariate moments of pairs of dimensions.

6.1 Related Work

The related work comprises approaches which perform a statistical analysis of user-defined subsets of the data, as well as approaches for supporting a high-dimensional data analysis in general.

Mathematically Describing Brushed Subsets – The interaction metaphor of brushing has established itself as proven standard approach to the identification of selected data subsets of interest (see section 2.4.1). Some approaches also perform a statistical analysis of the selected subset of the data in order to organize interesting queries [280] or to guide the user to similar parts of the data [102, 83]. However, while the intention of these approaches is to detect other potentially interesting parts of the data based on the description, our approach provides statistical summaries themselves to the user and ranks dimensions accordingly.

High-Dimensional Data Analysis – As surveyed in section 2.3, a variety of approaches addresses the non-trivial question how to explore truly high-dimensional datasets. Methods which are based on linear combinations of dimensions to project high-dimensional data to low-dimensional space are typically hard to interpret (e.g., Principal Component Analysis [138] or Projection Pursuit [79]). Other approaches visually represent the space of dimensions itself [7, 281], but they typically neither consider user-defined local features nor do they provide any numerical details as the work presented in this chapter. The same shortcomings apply to most approaches which rely on ordering dimensions with respect to certain metrics [283, 282, 201, 136]. Methods assessing potential projections by different user-dependent notions of interestingness [278, 230, 244] speed up the identification of interesting plots but typically require high abstraction skills from the user.

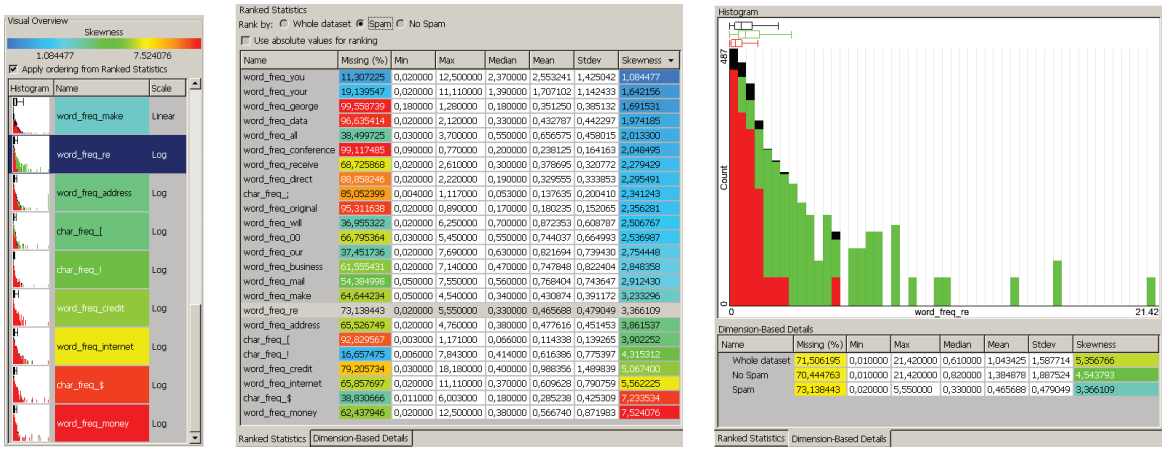
Most related to our approach, the Rank-by-Feature Framework [222] is designed to meet the Graphics, Ranking and Interaction for Discovery (GRID) principles: a) "study 1D, study 2D, then find features"; and b) "ranking guides insight, statistics confirm". The user may choose between several statistics displayed in a linked table for ranking preview visualizations of the dimensions. While this approach has proven suitable as an initial guidance to potentially interesting dimensions [223], it is of limited use when it comes to the focused analysis of selected data subsets of interest.

6.2 Quantifying Brushed Data Features

We now present our approach to visualizing, quantifying and comparing data subsets in the context of large numbers of dimensions.

6.2.1 The General Approach

Our approach distinguishes the subsets as defined by user queries (and also the whole dataset as a "special" subset) and restricts all statistical computations to the according and valid entries of the data. It considers an arbitrary number of dimensions, as selected by the user. All results are automatically updated whenever a query and thus the underlying subset changes (e.g., when the user brushes a linked view), hence providing full linking to all other views. The basic setup consists of three coordinated parts which support different tasks.



(a) (b) (c)

Figure 6.1: The 1D case showing word and character counts for spam classification of e-mails. (a) The Visual Overview displays mini-histograms, box plots, and color-codes the dimensions according to the current ranking criterion "Skewness". (b) The Ranked Statistics list the results of a user-defined set of statistical moments for each dimension. (c) The Dimension-Based Details provide a larger histogram, whisker plots for each layer, and the results of the statistical moments with respect to the active dimension for the whole dataset and each query.

The Visual Overview (Fig. 6.1a and 6.2a) displays small visualizations with little detail in order to provide an overview of all considered (pairs of) dimensions. Despite their small size, it has proven useful that also these mini-views highlight the subsets as defined by the user. The user may either manually arrange the dimensions or may automatically sort them according to the "active" statistical moment, defined by the Ranked Statistics (see below). If the results are comparable across all dimensions (i.e., if the range of that moment is independent of the scaling of the data), the user may visualize the differences by mapping the results to color. The according transfer function is scaled between the smallest and largest result of the respective statistical moment, unless a "natural" range exists (e.g. -1 to +1 for correlation coefficients).

Ranked Statistics (Fig. 6.1b and 6.2b) are structured as table with (pairs of) dimensions as rows and the respective moments as columns. It shows the results, which are optionally computed for the whole dataset or for the subset defined by a query and is thus suitable for simultaneously quantifying one query with respect to multiple dimensions. The rows can be ordered by any column, denoting the respective moment as "active" which also determines the order and color-coding of the Visual Overview. If applicable, each column is color-coded in order to improve the comparability.

Dimension-Based Details (Fig. 6.1c and 6.2c) refer to a single (pair of) dimension(s), which is selected in any of the other parts. The purpose is twofold: First, this part provides a larger visualization with more detail. Second, it displays a table similar to the Ranked Statistics with columns being the selected statistical moments. The difference is that the rows represent the results for subsets defined by the various queries (plus one row referring to the whole dataset) for the active (pair of) dimension(s), allowing for direct comparisons of

the characteristics for all queries.

6.2.2 1D Framework

We now explain how this general approach can be applied to analyze individual dimensions (1D case) and/or pairs of dimensions (2D case). The intention of the 1D case (see Fig. 6.1) is to look at the dimensions individually and the offered statistics are therefore univariate. The following set of different statistical moments allows to adapt the analysis to the user task and to the respective properties of the data:

- Minimum and maximum.
- Mean and median.
- Quartiles (1^{st} and 3^{rd}) and standard deviation.
- Trimmed mean and trimmed standard deviation for robust statistics: omits the smallest and largest 10% of the values.
- Skewness, kurtosis and normality: describe and quantify the deviation from normal distribution.
- Entropy: rises with increasing uniformity of the data distribution.
- Number of unique values.
- Value of the biggest gap.
- Percentage of missing entries.

The related visualization approach employs well-known histograms and box plots [171] to show the distribution of each dimension. The Visual Overview (Fig. 6.1a) consists of a list of dimensions, where each row contains a small box plot drawn above a histogram, which also highlights all subsets as defined by queries. In order to support multiple queries, which are not necessarily disjunctive, the results of the queries are drawn on top of each other with the "active" subset drawn in front. Furthermore, an attempt is made to determine whether the Y-axis of the histograms should better be scaled linearly or logarithmically in order to guarantee meaningful visualizations, also for distributions where the majority of values lies in a very narrow range – of course, the user may manually override this setting. The Dimension-Based Statistics display a large histogram and a box plot for each query (Fig. 6.1c).

6.2.3 2D Framework

Apart from analyzing dimensions separately, users are typically interested in relationships between multiple dimensions. This is true on a global scale (e.g., identifying groups of similar dimensions) and also applies to local features like individual clusters. In order to support such tasks, the 2D framework (see Fig. 6.2) allows for exploring all combinations of assigned dimensions. Therefore, the investigated items are pairs of dimensions. Concerning the handling of missing data, this implies that only entries are considered valid if they are present in both respective dimensions. Due to the symmetry of all employed techniques, it turned out

(a) (b) (c)

Figure 6.2: The 2D case comparing pairs of word and character counts for the spam classification dataset. (a) The Visual Overview shows a part of the scatterplot matrix, the background color-coded by the percentage of missing entries and the pair of the word counts of "you" and "all" currently highlighted. (b) The Ranked Statistics list the results of the bivariate statistical moments for each pair of dimension, with "Missing (%)" being the current ranking criterion. (c) The Dimension-Based Details show a larger scatterplot and compare the results with respect to the active pair for the whole dataset and each query.

to be sufficient and fosters the overview to maintain only one pair per combination (i.e., for two dimensions X and Y, either (X, Y) **or** (Y, X)) and to omit all pairs of any dimension with itself.

Any symmetric bivariate statistical moments are suitable for the 2D approach. The currently available moments comprise Pearson's correlation coefficient and Spearman's rank correlation coefficient [185] (plus the percentage of entries considered as missing). While the first one is appropriate for describing linear relationships, the latter provides more robustness and the ability to detect also non-linear dependencies at slightly higher computational costs. Both coefficients range from -1 to +1 independent of the scaling of the data and are thus suitable for coloring.

Relationships between two dimensions are usually visualized with 2D scatterplots, which are also used in our case. While the Dimension-Based Details display a scatterplot with higher resolution showing more details of a selected pair (Fig. 6.2c), the Visual Overview arranges the potentially large number of plots as a scatterplot matrix (Fig. 6.2a). Due to exploiting symmetry as explained above, a single plot is drawn for each pair, which reduces the matrix to a triangle and leaves space for printing the names of the dimensions. If the extents of the matrix exceed the available space, it is scaled down to a certain minimal size, before scrollbars are shown. However, the plot beneath the mouse cursor is always zoomed smoothly to its original size. The exact layout is based on a linear order of the dimensions (like in the 1D case). The user can specify this order manually or adopt the order of the Ranked Statistics.

Unlike the 1D case, automatically obtaining an order from ranking dimension pairs is not straightforward and ambiguous. After evaluating several strategies, we employ the following algorithm for this task: First, those two dimensions are selected of which the pair achieves

the highest ranking, which specifies the topmost plot. Afterwards, the algorithm selects the dimension, which produces a row of the matrix including the pair with any already assigned dimension, which has the highest ranking. This latter step is repeated until all dimensions have been assigned, thus constructing the matrix line by line. The benefit of automatically ordering the matrix is that similar dimensions tend to be placed close to each other, indicating groups of dimensions more directly.

6.2.4 Further Aspects of Our Approach

The approach described in this chapter has been realized in the context of the system Visplore as briefly described in section 5.3. A key aspect of Visplore with regards to local features is the support of multiple queries, which are defined by composite brushing in various linked views. Both the 1D and 2D framework introduced in this chapter implement the multi-threaded architecture as described in chapter 3 to support datasets with millions of entries. Visplore explicitly allows for denoting single values as missing, which are expected to be omitted for all visualizations and computations.

6.3 Demonstration

This section briefly illustrates a potential workflow with our approach by analyzing a dataset that has been used for classifying e-mails as spam or no spam. The dataset is based on 4601 e-mails. It contains the relative frequencies of certain words and characters in the respective message and whether it is regarded as spam, summing up to 57 dimensions. It has been obtained from the UCI Machine Learning Repository [9] and originates from the Hewlett-Packard Labs, where employees collected and classified e-mails in order to build a personalized spam-filter. The goal of this case study is to show that our framework supports the task of assessing words and joint occurrences of words with respect to the relevance regarding spam classification. Note that counts of zero are treated as missing values.

As first step of the analysis after importing the dataset, two queries are created in order to select all e-mails, which are classified as spam (red) and no spam (green), respectively. This is accomplished by interactively brushing a linked view for visualizing such categorical dimensions (see Fig. 6.3).

As the next step, all dimensions related to counts of words and characters are assigned to the 1D case (see Fig. 6.1). The histograms show that most dimensions are distinctly exponentially distributed, i.e., most counts have many small and very few large numbers of occurrences. Therefore, the Y-axes of most histograms are logarithmically scaled in order to make also small occurrences visible. Moreover, many dimensions have a lot of missing

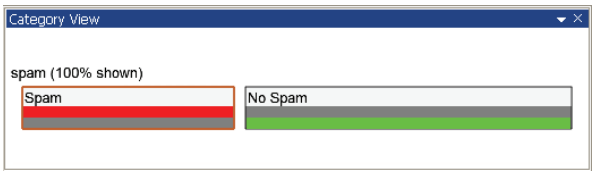


Figure 6.3: Brushing mails classified as spam (red) and no spam (green) in a linked view for visualizing categorical data.

values, which means that these words do not occur at all in many messages. When looking at the histograms, the distinctive coloring of spam and no spam messages reveals that the distribution per subset is quite different for individual words and characters: Some clearly occur more often in one class while for others, the distribution is more or less equal. Ranking the Visual Overview by the average number of occurrences of spam mails provides a very approximate ordering with respect to the likelihood to indicate spam. Picking one word ("re") as an example of a dimension, where the overview suggests good indication properties, the Dimension-Based Details (see Fig. 6.1c) confirm this assumption (e.g., by different box plots and mean values). However, due to the high degree of missing values, it is obvious that single words and characters will not be sufficient for a good classification.

Therefore, joint occurrences are analyzed in the 2D case, where words with promising indication properties are assigned to (see Fig. 6.2). Because some words occur together only rarely, the percentage of missing data is mapped to color in order to indicate the significance for each combination. As an example of a comparatively frequent pair (see Fig. 6.2a), inspecting the combination of the words "you" and "all" in more detail (see Fig. 6.2c) shows that this pair is missing in 78% of the messages not considered as spam, but only in 40% of the spam messages. In other words, this dataset suggests that encountering both words in one e-mail significantly increases the likelihood for being spam, though it is of course no proof on its own - for this, more pairs would need to be considered together. Furthermore, the Dimension-Based Details also show that the number of occurrences for "you" and "all" are much more correlated for e-mails being no spam (with $\sim 47\%$ according to Pearson and $\sim 56\%$ according to Spearman). However, probably most interestingly, the scatterplot shows some very distinct "needles", where several messages obviously have perfectly linearly correlated counts of these two words. Such structures are impossible to explain without further knowledge about the e-mails and raise questions regarding the quality and authenticity of the data. If this can be justified, these features suggest the existence of multiple classes of mails, which could be the starting point of a more in-depth analysis.

6.4 Conclusions and Future Work

In this chapter, we have introduced an approach for quantifying and comparing multiple subsets of a dataset by computing and ranking univariate as well as bivariate statistical moments with respect to an arbitrary number of dimensions. The subsets are defined by interactive brushing in linked views. The 1D case supports analyzing multiple dimensions separately, while the 2D case reveals relationships between dimensions. Like the Rank-by-Feature Framework by Seo and Shneiderman [222], our approach is well suited for conveying a quick overview about global properties of the dimensions at an early stage of analysis. However, the aspect of linking the approach to other views significantly extends its applicability also to later stages of analysis – e.g., computing statistics after deselecting identified outliers, characterizing detected clusters, or comparing various categories to each other. It turned out that using well-known statistics leads to faster understanding and acceptance of the approach for domain experts, and extending the set of offered statistics is easily possible.

Concerning scalability, the most important goal of our approach is to facilitate an analysis of high-dimensional data. However, there is a certain practical limit concerning the number of simultaneously shown dimensions. Due to the quadratic increase of dimension pairs, this limit is significantly lower in the 2D case. Our experience shows that approximately 35 to

40 dimensions can reasonably be handled in the 2D case, while the 1D case also works well for a few hundred dimensions. With regards to the number of data items, the scalability is mainly limited by the effort to compute the offered statistics as well as by the limited visual scalability of the preview visualizations. The involved computations are fast even for millions of data entries and the histograms employed by the 1D case also scale very well due to binning the data. However, as most scatterplots, the preview visualizations in the 2D case suffer from overplotting already for a few million data items (see chapter 4).

We see at least two directions for potential future work. First, more work would be helpful on how to automatically extract and quantify the characteristics and even semantics of brushes. Second, exploring really high-dimensional datasets with several hundreds or even thousands of dimensions is still a big challenge. As our approach also generates tables with (pairs of) dimensions as rows, applying well-known visualization techniques for multivariate data like parallel coordinates to such tables could be an interesting start.

Chapter 7

Interactive Visual Validation of Regression Models

Strict emission rules and steadily increasing demands on performance force car manufacturers to constantly improve the design of powertrain systems [24]. Different types of numerical simulations have thus become a key technology to analyze complex systems like engines. Especially in early stages of the design process, 1D Computational Fluid Dynamics (CFD) simulations are widely used. Compared to very time consuming 3D CFD simulations, 1D CFD simulations are magnitudes faster and produce much less data. This enables to study the design space by running thousands of simulation runs for different locations of the parameter space. Matkovic et al. [174] describe how interactive visualization can support the analysis of such data.

Some tasks, however, require mathematical models that can make predictions in real-time. When testing strategies for Engine Control Units (ECUs), for example, their behavior is analyzed during a simulated drive, which requires simulating the physical behavior. As even 1D CFD simulations are far too slow for such real-time tasks, manufacturers use *surrogate models* instead which are based on statistical regression rather than on physical equations [114].

The identification of surrogate models requires known results at sampled positions of an input parameter space for training and validation. These results come from measurements or from 1D CFD simulations. A particular model is trained to predict one result (e.g., torque) given values for particular attributes (e.g., speed and load). While the process of training a model requires no interaction, most types of regression models have numerous parameters (e.g., kernel type and tolerance values for support vector regression [106]) that must be set before. The complex interplay between these parameters makes finding good training parameters a challenging task. Different models may also perform best for different subsets of input attributes. Sometimes, implausible data must be filtered before the training.

Before making critical decisions based on the predictions of a surrogate model, it is therefore very important to validate its quality and to possibly improve it in further iterations. This validation involves comparing known results to predictions by the model [217]. Besides looking at criteria that can be derived automatically (e.g., the maximal residual), it is essential for application experts to get a feeling for the model itself and to be able to analyze its behavior also for regions not covered by training data. For these reasons, a fully automatic workflow for model validation is insufficient. Instead, it must be complemented by an interactive visual approach to increase the confidence in the overall process based on the domain

knowledge of the engineers.

Based on interviews with application experts and observations of their workflow, we derived the subsequent set of *design goals* for an interactive visual approach for the validation of surrogate models:

- Relate known results to predictions of models.
- Support a quick identification of deviating regions.
- Convey a feeling of the behavior of a model around any point in space.
- Allow a comparison of multiple models.
- Scale to models with a large number of input parameters.
- Be highly interactive without perceivable delays.
- Be tightly integrated into the workflow of model identification by restricting the analysis to an arbitrary subset of validation data and to update the visualization immediately at modifications of the model.

Although the application background of this chapter is the development of powertrain systems, the identification and the validation of regression models is of general importance in many application domains [234]. The problem can be formulated as assessing the suitability of a multi-dimensional scalar function $y = f(X)$ to approximate known values of y at a set of different points for X, where y is a scalar value (called the dependent variable) and X is a point in n-dimensional space (referred to as independent variables). While combined visualizations of the function graph and known results are very common in statistics for one-dimensional regression models, a detailed analysis of higher dimensional models is challenging. The issue of scalability with respect to the involved dimensionality thus plays an important role for our task.

The *contributions* of this chapter are as follows:

- A *design study* of HyperMoVal, an interactive visualization technique for validating regression models based on a combined visualization of n-dimensional functions and known validation data.
- A tight integration of HyperMoVal within an *interactive workflow* for iterative identification and validation of regression models involving multiple linked views.
- An *evaluation* of the proposed techniques within the application context of engine design based on a case study of an exemplary workflow and the report of user feedback.

7.1 Related Work

Numerous visualization techniques address the important issue how to visualize multivariate data [96]. Most of these techniques operate on generic n-tuples, for example parallel coordinates [127] and scatterplot matrices [43].

Besides such general purpose techniques, various approaches have been proposed to visualize multi-dimensional scalar functions $y = f(x_1, x_2, ..., x_n)$. A direct visualization is trivial

for n = 1 using line graphs and n = 2 using surface plots. The case n = 3 is typically addressed using volume rendering and isosurfacing. However, larger values of n are challenging. Most visualization approaches thus reduce the number of parameters to 1, 2, or 3 per plot by assuming specific values for the rest. These values can typically be changed by animation or by interaction.

Worlds within Worlds [69], for example, uses a hierarchy of nested coordinate systems. At each level of the hierarchy, the origin of the coordinate system specifies up to three parameters for all further levels until the number of independent variables is reduced to support a direct visualization of the function. A related approach by Mihalisin et al. [182] nests axes hierarchically to plot function values against all combinations of a sampled subset of the parameter space. Jayaraman and North propose a focus+context visualization of multi-dimensional functions [131] based on a radial layout of slices. Each slice shows the behavior of the function as sampled along rays emanating from a focal point into a certain direction for one independent variable.

HyperSlice by van Wijk and van Liere [261] shows all 2D orthogonal slices of a function around an n-dimensional focal point. Each slice represents the function by varying two parameters while assuming the values of the focal point for the rest. This is done for all pairs of variables and the resulting plots are arranged using a matrix layout. Dos Santos and Brodlie extended the idea of HyperSlice to three dimensions to what they call HyperCell [57]. Instead of using a fixed matrix layout, they provide a graph interface that supports the creation of 1-, 2-, or 3-dimensional plots (called cells) and they support multiple focal points. Recently, Nouanesengsy et al. [193] proposed an approach to analyze multi-dimensional scalar functions based on projecting line segments into 2D plots. The key idea is to plot the distance of each line segment from user-defined points against the respective function values.

All of these techniques are useful for understanding certain aspects of multi-dimensional scalar functions, and particularly HyperSlice served as an important and widely accepted starting point for our work. However, the goal of these approaches is to visualize a function as such, not to validate it as an approximation model by means of known data. As motivated above, a validation requires a direct comparison of the approximation model to results from measurements or physically-based simulations [217].

Moreover, the identification of regression models based on techniques from machine learning is typically an iterative process [19]. Each iteration involves parameter definition, training, and validation. Supporting this entire workflow is an important topic which goes far beyond the mere visualization of functions. Recent work shows significant progress towards supporting the process of model building using interactive visualization, but this work is restricted to particular models like clusters [188] or linear trends [98]. There is no approach to support the identification and validation of general regression models according to the design goals listed above.

7.2 Interactive Model Validation

This section introduces HyperMoVal as our approach to an interactive visual validation of surrogate models. We first describe the combined visual encoding of model and validation data before discussing the supported interaction techniques.

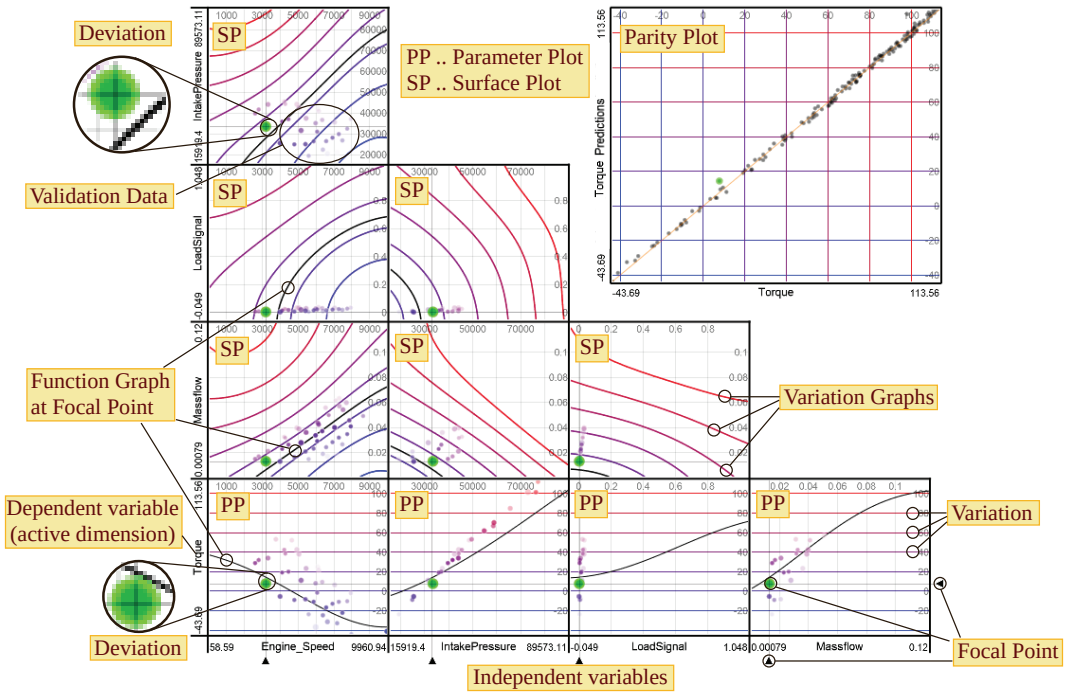


Figure 7.1: The layout of HyperMoVal for a real model predicting torque given four parameters. The focal point F is set to a validation data point with a significant deviation. The matrix contains all paraxial 2D slices at F in the 5D model space.

7.2.1 Visual Encoding

The key idea of our approach is to provide a combined visualization of a regression model and known validation data in order to assess the match between both. The model is an analytically given scalar function with n independent variables and the validation data is given as an arbitrary number of n+1-tuples, i.e., a value for each parameter of the function and one for the result. In order to visualize such potentially high-dimensional data, we layout multiple projections to low-dimensional space as paraxial slices in a similar way as HyperSlice [261]. This approach treats all dimensions equally and supports the creation of plots which are familiar to engineers. An implication is to maintain a *focal point* F as n+1-tuple, which defines values $F_i, i \in \{1..n+1\}$ for all dimensions not shown by a particular plot. F may take any position inside an n+1-dimensional hyperrectangle S representing the space considered for visualization. S is defined as the Cartesian product of intervals defined for each of the n+1 dimensions. S is initially set to contain all values of the validation data, but its extents can be modified by the user (see section 7.2.2).

Each pair of dimensions defines one plot (see Fig. 7.1). The result dimension of the function specifies the y-axis of the bottom row of the matrix. These plots (called *parameter plots*) thus display the explicit function graph for all independent parameters. All other plots (called *surface plots*) show pairs of independent variables, where the function is represented as contour at the iso-value specified by F. Only the lower triangle of the matrix is displayed for performance reasons and to gain free space for other purposes.

In addition to the function graphs of the model, each plot also displays projections of the validation data as points. In order to support a quick identification of badly approximated

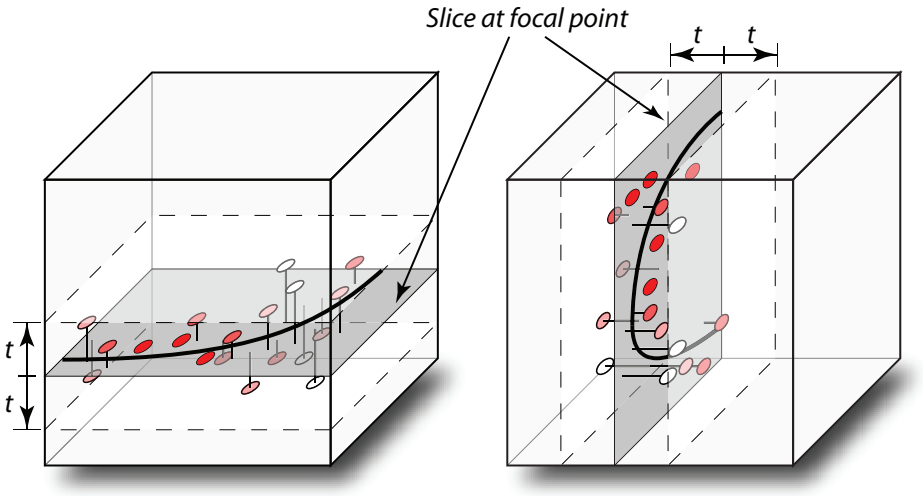


Figure 7.2: The region around two slices in which points are considered relevant for the respective plot. The color intensity depends on the distance to the slice.

regions, the *size* of each point reflects its absolute residual to the prediction by the model at the particular position in parameter space. Large points indicate regions with a significant deviation whereas small points suggest a good match. Although the precision conveyed by size is lower than for other visual attributes, it is sufficient for the purpose of assessing the prediction quality.

7.2.1.1 Task-Specific Point Relevance

As each plot is a slice through the visualization space S at F, it is important to consider which points should be displayed by a plot. A precise assessment of the fit by a model is only possible for validation points on that slice. One approach could thus hide all other points. In this case, at most one point will typically be visible. An important exception, though, is validation data which is structured as – possibly unequally spaced – n-dimensional grid. This is a common case in the design of experiments (e.g., full-factorial designs). Provided that F is set to a point of the grid, parameter plots then show all points in the same line in the respective dimension. 2D surface plots, however, require a precise match in the result dimension and typically display F only (this is different for 3D plots as discussed below).

In general, it is often reasonable to additionally display points close to (but not on) the slice of a plot, e.g., to provide context information for navigation. It may even be necessary to show all points in order to assess the coverage of the parameter space by validation data like in a normal scatterplot matrix. For this reason, HyperMoVal supports a task-specific definition of what is considered a *relevant range* around each slice (see Fig. 7.2). Formally, let P denote a point inside the n+1-dimensional hyperrectangle S and let V be a 2D visualization of the dimensions x_j and x_k, then the relevance r of P with respect to V is defined as

$$r_V(P) = 1 - Max_{i \in \{1..n \setminus x_j, x_k\}} \left(\frac{|P_i - F_i|}{|S_i|} \right)$$

where S_i denotes the length of S in the i-th dimension. After computing $r_V(P)$, a linear mapping determines the saturation with which P is drawn. This mapping is controlled by a threshold t with $0 \leq t < 1$ so that $r_V(P) \leq t$ means white (i.e., the background color) and

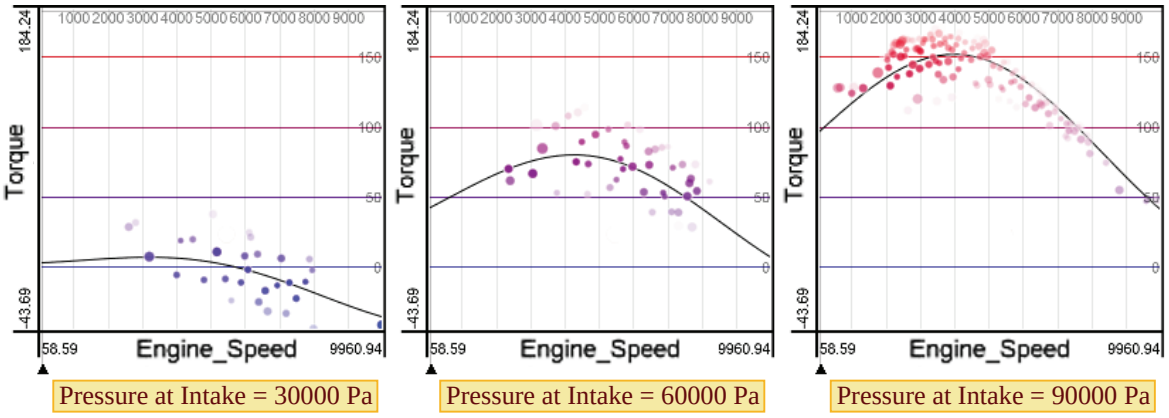


Figure 7.3: Altering F in one dimension changes the function graph and the relevance of data points for all plots not displayed in that dimension. Continuous modifications make the points fade in and out smoothly around the graph.

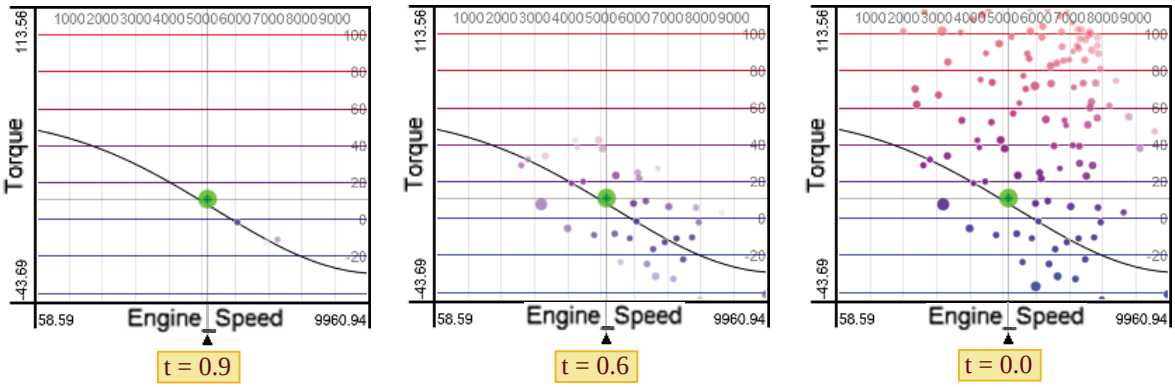


Figure 7.4: Altering the relevance threshold t changes the shown subset of data points around the visualized slice.

1 means maximal saturation. The transition conveys the distance to the slice and generates a smooth fading effect when changing F, as illustrated by Fig. 7.3. Modifying t enables to define relevance depending on the task (see Fig. 7.4): values close to (or at) 1 show only a narrow range around the slice to precisely support a visual assessment of the fit. For values close to 0, our approach resembles a normal scatterplot matrix.

7.2.1.2 Sensitivity Analysis

Another important task is to analyze the sensitivity of the surrogate model with respect to changes in single dimensions. To support this task, one dimension can be set active at a time, which has two effects: 1) each plot displays a family of function graphs (called *variation graphs*) for a user-defined number of equally sized steps along the displayed range of the active dimension; 2) the active dimension determines the hue of validation data points and variation graphs via a transfer function. By default, we use a transition from blue to red. This ensures a good luminance contrast to the white background and it is perceptually distinct from the modulation of saturation by point relevance. Users may also choose other transfer functions.

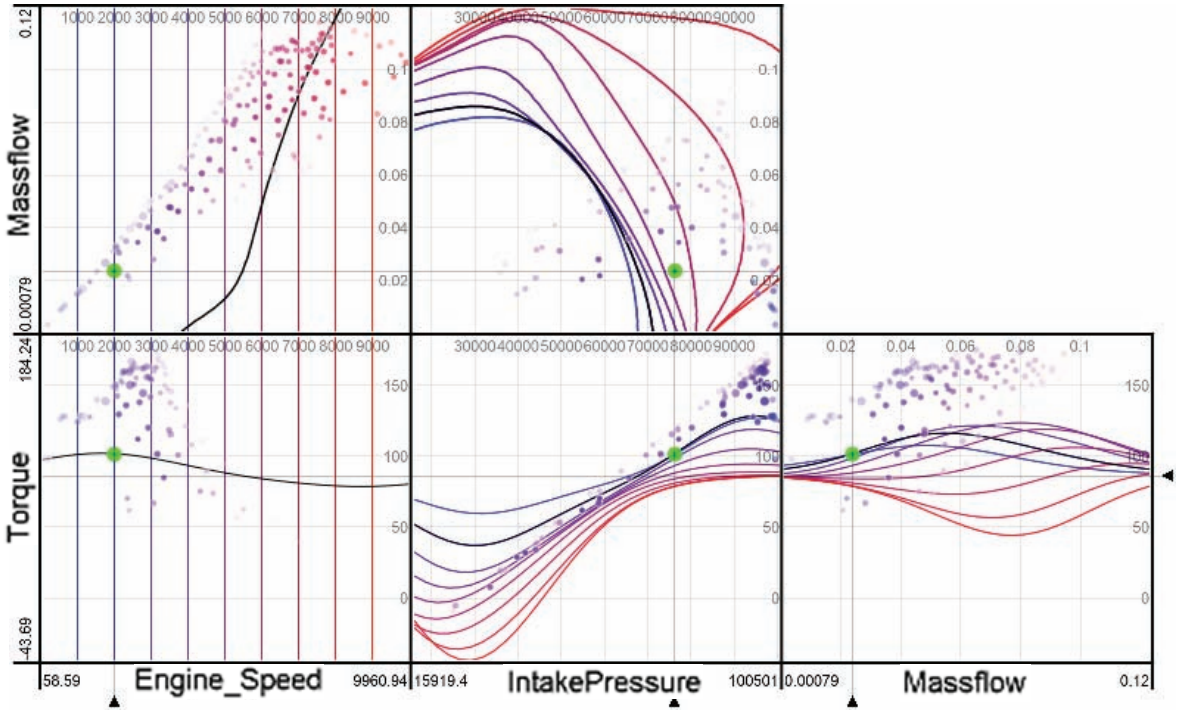


Figure 7.5: Varying the parameter "Engine Speed" generates a family of iso-contours and explicit function graphs.

Coupling variation and coloring has proven beneficial with respect to usability. The main reason is that grid lines of plots showing the active dimension define meaningful steps for variation graphs and may also be used as legend for the color coding. The most frequent – and default – case is to set the result dimension active, which generates multiple iso-contours of the same surface in each surface plot (as in Fig. 7.1). Setting a function parameter as active dimension generates multiple explicit graphs in all other parameter plots. It also shows contours at the same iso-value for multiple surfaces in each surface plot that does not display the active dimension (Fig. 7.5). Mapping the active dimension to hue emphasizes relations between points and graphs with respect to this dimension. It also reveals deviations when points are afar from graphs despite having the same hue.

7.2.1.3 3D Visualization

While all plots are described as 2D so far, surface plots optionally provide a *3D visualization* as well. In this case, the result dimension of the model is mapped to height (see Fig. 7.6). The validation data is shown as a 3D scatterplot of points which are scaled and colored as described for the 2D case. The computation of relevance excludes the result dimension in order to display points around the entire surface. Being less restrictive, more data points are thus shown in 3D than in 2D. The function of the surrogate model is represented in multiple ways: iso-contours are generated for F and all variation graphs of the active dimension as line strips similar to the 2D case, except that each contour is located at the height of the respective iso-value.

In addition, the surface of the surrogate model for F is either shown as wireframe or

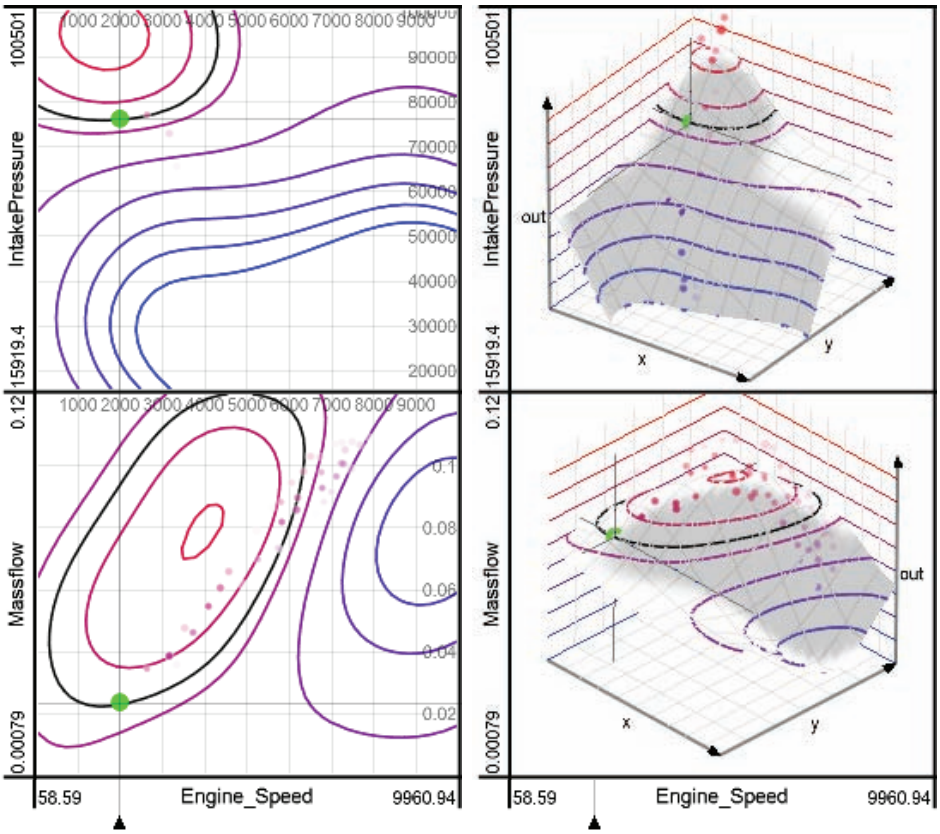


Figure 7.6: 2D and 3D visualizations of the same surface plots. Due to the additional visual dimension, the set of relevant points is typically larger in 3D.

as shaded, opaque surface. While the wireframe representation avoids problems caused by occlusion and thus keeps all validation points visible, opaque surfaces convey a better 3D impression at the cost of occluding points either above or beneath the surface – depending on the viewing position. When switching between 2D and 3D, a smooth transition of the view point avoids change blindness in a similar way as described by Elmqvist et al. [65]. The main reason for integrating a 3D visualization is the great popularity of this representation with engineers in our target domain. Initial observations have shown that 3D plots are particularly used for assessing models with few parameters and for checking physical relationships between parameters. 2D surface plots are rather used for studying abstract relationships for high-dimensional models.

7.2.1.4 Parity Plot

Differences between predicted and known results are also shown in a separate scatterplot (see Fig. 7.1). This plot maps known results to the X-axis and predicted results to the Y-axis for each point of the validation data. Such plots are called *parity plots* and are common in many fields of science and engineering. A straight line at the main diagonal indicates a good match whereas deviations to either side convey the amount and the sign of residuals. The thickness of the covered area around the main diagonal provides an overview of the overall fit and outliers remain visible. Both types of information could also be conveyed differently

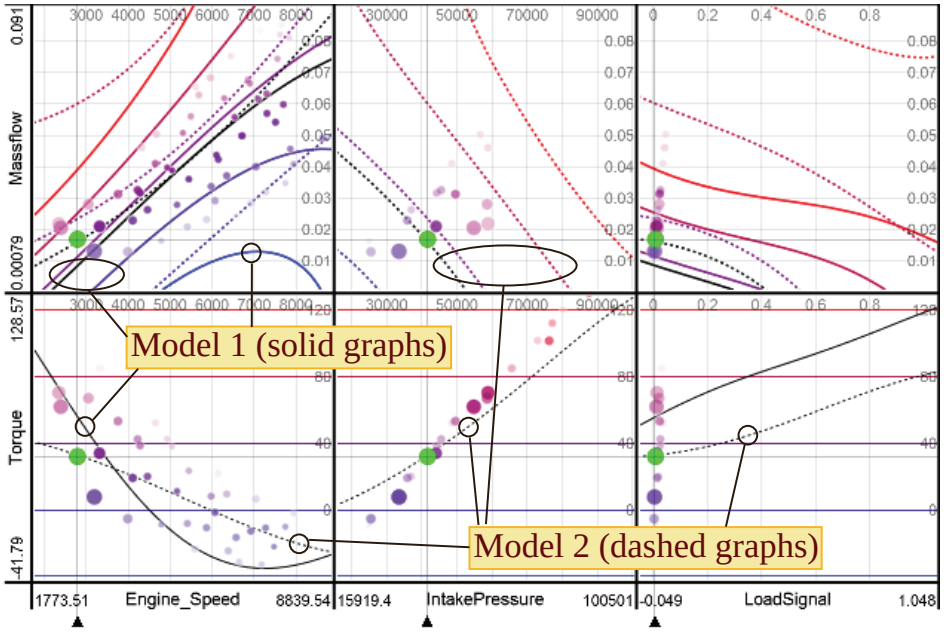


Figure 7.7: Comparison of two models predicting torque. The models are discriminated by the line style (dashed and solid) and show a very different prediction quality at F. "Intake-Pressure" is a parameter of Model 2 only.

(as shown in later sections). The reasons for integrating parity plots are the equal scaling of both axes and the popularity of this plot with experts in our target field of application.

7.2.1.5 Comparison of Multiple Models

Another design goal concerns the *comparison* of multiple regression models predicting the same result dimension. The independent variables may be different, in which case our approach represents the union of the independent variables of all visualized models. Each plot (except the parity plot) shows function graphs or iso-contours for each model that covers the dimensions of both axes. Different line stippling is used to discriminate the graphs of multiple models (see Fig. 7.7). Deviations, however, will in general be different for multiple models. In order to avoid overloading the visualization, the point size thus refers to the residuals with respect to one active model while disregarding all others. Analogously, the parity plot is only shown for the active model and 3D surface plots show the – wireframe or opaque – surface of the active model only. For these reasons, other models are rather context information when assessing the match between the active model and the validation data. However, visualizing multiple models is particularly useful when comparing their behavior outside the space covered by validation data (e.g., when assessing the plausibility of extrapolations).

7.2.2 Interaction

Interaction is an important aspect of HyperMoVal to support different tasks. *Modifying the focal point* F enables to navigate the visualization space S. We discriminate local and global modifications: *local modifications* affect a subset of the dimensions at a time. They involve altering the value of a single dimension by dragging handles at the border of the matrix

or altering two dimensions by moving a crosshair representing F in any 2D plot. *Global modifications* define values for all dimensions at once by setting F according to a particular point. This is done by clicking on a visible validation point in any 2D plot including the parity plot. It supports a quick navigation to points with a significant residual.

As another type of interaction, the extents of S can be modified individually for each dimension. Similar to the Prosection matrix [258] and dynamic filtering [2], narrowing the range of one dimension removes all validation points outside this range in all plots. Changing the range also allows the user to analyze every part of the function in as much detail as necessary or to scroll across function graphs. Extending ranges to contain space outside the region covered by validation data is crucial in our context to assess the plausibility of *extrapolations* by a model.

Further interactions which are not discussed in detail involve changes of the viewpoint in 3D plots, altering the threshold t as discussed in section 7.2.1, and setting functions and dimensions to be active. When validating models with many parameters, it is helpful to temporarily hide certain independent parameters and thus gain more space for the rest.

7.3 Integrated Workflow for Model Identification

HyperMoVal as described in section 7.2 can be regarded a design study that is conceptually independent of properties of a surrounding software system. A stand-alone implementation of HyperMoVal already fulfils all design goals except the last as listed above.On the other hand, it is not always practical for a single view to cover all potentially relevant tasks during a complex workflow like model building. Fortunately, linking multiple views is agreed to be a solution to this problem. This section therefore describes the integration of HyperMoVal within a system of multiple linked views. We also discuss how to use different views to support the entire workflow of model identification. It should be emphasized, however, that the focus of this chapter is on model validation while many aspects related to model identification are up to future work (see section 7.6).

Our integration is based on two concepts: 1) defining arbitrary subsets of validation data in any view and 2) representing predicted results and residuals of a particular model as derived data attributes. Besides HyperMoVal, the types of involved views may include most standard visualizations for multivariate data, for example scatterplots and parallel coordinates. Their purpose is to visualize the validation data and to support the definition of arbitrary subsets thereof based on view-specific brushing techniques like rubberband- or lasso-selections. HyperMoVal supports restricting the set of validation data points to any subset defined by brushing other views. This enables to filter validation data that is implausible (e.g., incorrect measurements) or undesirable due to design constraints. Unlike the filtering offered by HyperMoVal, such selections may include dimensions which are not covered by any model and they may be based on composite queries of arbitrary complexity [274].

As second concept, predicted results and residuals of a particular model are added as additional data dimensions for each point of the validation data. Such *derived data columns* are available for view parameterization like any attribute of the validation data itself. This offers a variety of possibilities: for example, residuals of multiple models may be compared using parallel coordinates (Fig. 7.8a). They may indicate the amount of deviation in a scatterplot relating model parameters (Fig. 7.8b). Other views may provide statistics as quantitative results, e.g., the maximal and the average prediction error of a model for a certain subset

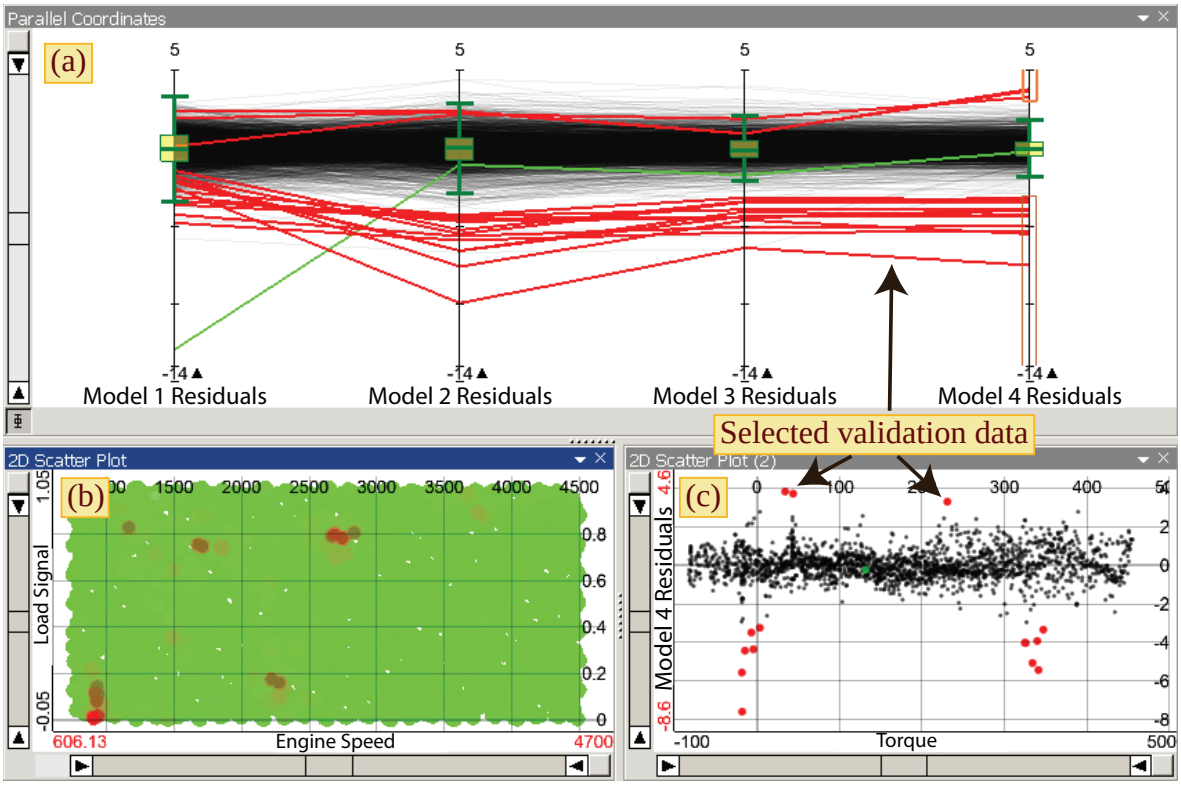


Figure 7.8: Linked views of derived model attributes: (a) comparing the deviations of different models by brushing large residuals; (b) mapping absolute deviations of one model to color in a 2D representation of the parameter space; (c) plotting residuals (Y-axis) against known results (X-axis).

of the validation data (see chapter 6). Derived columns may also be used as criterion for selections as discussed above.

Validation is only a part of the overall workflow of model identification [217]. In our application context, the first step is the definition of all information that is necessary for training the model. This involves selecting dimensions as input parameters, specifying model and training parameters, and choosing the training data itself. After the (automatic) training, the model needs to be validated and potentially compared to other models as supported by our approach. The goal of this validation is to determine whether a particular model is appropriate, or whether further refinements are necessary. In the latter case, an additional iteration starts by changing the set of input dimensions, modifying training parameters, and performing another training [19].

Our approach to support this highly iterative process is based on integrating parameter specification, training, and validation within a single tool. This has two benefits: 1) brushing various views as discussed above can also be used to define the set of training data interactively, e.g., to filter wrong results based on domain knowledge of the user. 2) changing a certain model by executing a new training immediately updates the information related to this model in all views. This includes function graphs in HyperMoVal, derived attributes like residuals in all views, and it triggers a re-evaluation of selections which are based on derived attributes of the model. The advantage of this tight integration is the speed at which different settings of

training parameters can be tried and compared to each other, which may reduce the necessary time of the overall workflow significantly.

7.4 Implementation

HyperMoVal and the workflow described in section 7.3 have been implemented in Visplore, a system for visual exploration, which offers different visualizations that can be linked by ad-hoc selections and derived data columns (see section 5.3). All parts are written in C++ and use OpenGL for rendering.

HyperMoVal supports immediate visual feedback during continuous user interactions like changing the focal point F. For this purpose, it implements the multi-threading architecture as described in chapter 3. When drawing function graphs, a dedicated visualization thread samples the model progressively. This provides an immediate preview while generating the final image including variation graphs in each plot may take a few seconds on standard PCs for models of more than two parameters. Each plot also layers the visualization in order to prioritize the visualization of F and to cache and reuse visual results in image space.

Internally, models are objects which implement an interface that allows the integration of any type of regression model. Currently, our implementation supports support vector regression (SVR), and it uses the library LIBSVM [37] both for model evaluation and training. The user interface for defining training parameters is a simple dialog.

7.5 Evaluation

To evaluate our approach on multiple levels [187], this section first describes an application scenario to illustrate a typical workflow in car engine design. We then report general feedback collected by interviewing application experts of the target domain. This evaluation has been done in collaboration with three experts in the field of engine design, whose educational background is in mechanical engineering and industrial mathematics. As employees of AVL List GmbH, a company providing hardware and software for the development of powertrain systems, their responsibility is partly customer support and training, and partly the development of the simulation core. All of them are experienced users of our approach who have been testing, using, and training it for months.

7.5.1 Application Scenario

The goal of the scenario is to identify a surrogate model predicting torque based on the results of 400 simulation runs of a real-world car engine. This example uses the same data for training and validation. Knowing that torque is primarily a function of engine speed and load signal, the first step is to check whether a 2D model on these attributes already provides sufficient accuracy. Statistical summaries (first row of Fig. 7.9e), however, indicate a large mean deviation and a substantial maximal error. An analysis in HyperMoVal (Fig. 7.9a) reveals an outlier in the data. The engineer explains this point as non-converged simulation run and decides to exclude it from further steps. Besides the outlier, large points in regions near steep changes in torque generally reveal the model as insufficiently accurate.

In the next iteration, the engineer decides to increase the complexity of the model by adding the pressure at the intake manifold (p0IM) as third parameter which is known to

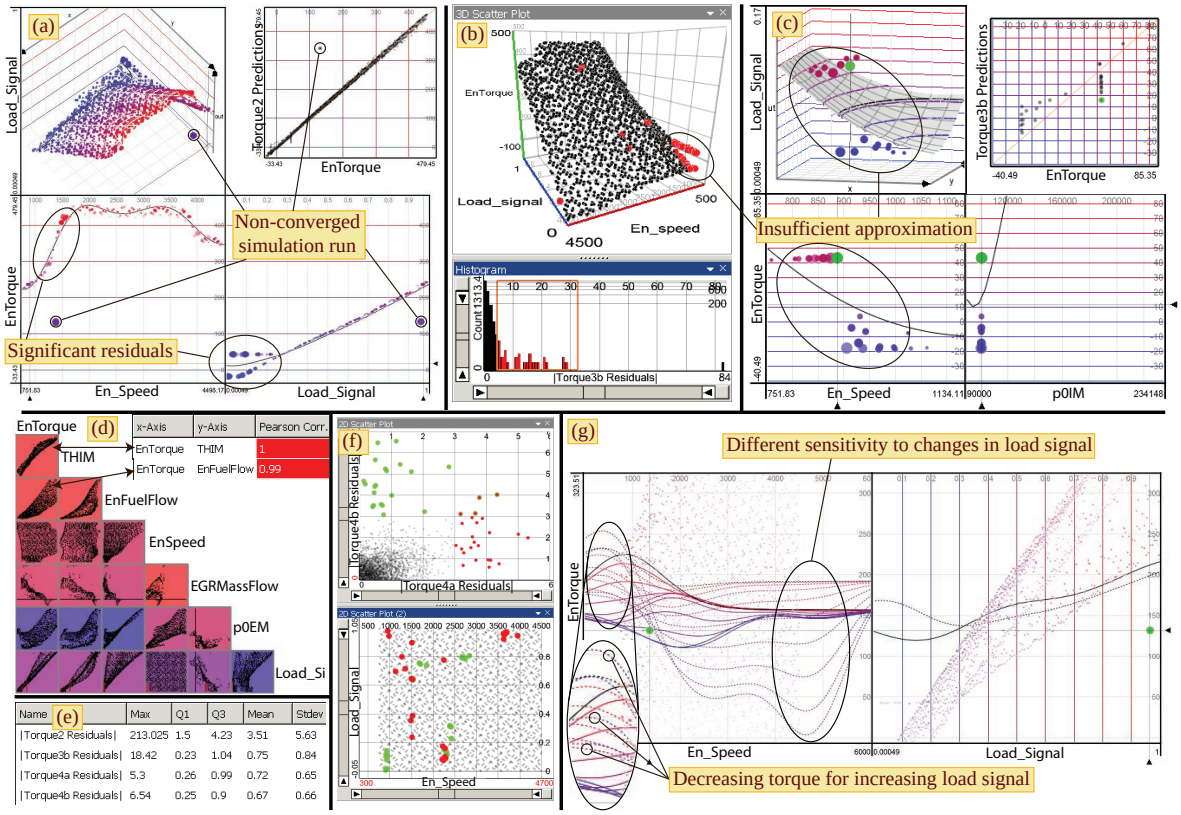


Figure 7.9: An exemplary workflow for model building. (a) Assessing a 2D model also reveals implausible validation data; (b + c) a 3D model still does not sufficiently cover a jump in torque for low values of speed; (d) ranking potential input parameters by means of correlation to the result; (e) quantifying multiple models with respect to their residuals (in Nm); (f) comparing residuals for two model candidates; (g) comparing the candidates with respect to the plausibility of extrapolations.

impact torque. While this new model turns out to be much better, statistics still show a considerable maximal deviation (second row of Fig. 7.9e). Brushing absolute residuals larger than 5 Newton meters (Nm) in a histogram (Fig. 7.9b) reveals that the main problem is a jump in torque at low values for both speed and load. As this insight still does not explain, why the match is bad in this region, the engineer uses HyperMoVal to analyze the local model behavior in detail (Fig. 7.9c). This shows that the gradient of the model is not sufficient to approximate the sudden jump in torque. A potential explanation is that p0IM does not contribute information for this region since the values for p0IM are very similar as shown by the bottom right plot of Fig. 7.9c.

The engineer decides to add one more dimension as model parameter. Since there are multiple physically meaningful candidates, ranking them by the amount of correlation to torque facilitates the selection (Fig. 7.9d). For this purpose, the engineer brushes the validation points in the mismatching region; he then opens a scatterplot matrix that is ranked and colored by the correlation of the selected points to torque as described in chapter 6. Having correlation coefficients close to 1, the temperature at the intake manifold (THIM) and fuel mass flow (EnFuelFlow) turn out to be equally suitable candidates.

In order to try both variants, the engineer creates two further models: The model "Torque4a" has THIM as fourth parameter while "Torque4b" depends on EnFuelFlow. The statistical quantification shows that both models are better than the previous ones in particular with respect to the maximal error which can now be considered sufficiently small (Fig. 7.9e). Brushing residuals greater than 3 Nm for both models in a scatterplot shows an error distribution that is slightly more clustered for model "Torque4b" (green points) in a linked view plotting speed against load signal (Fig. 7.9f).

As final step, the engineer uses HyperMoVal to compare the plausibility of both models (Fig. 7.9g) outside the range covered by training data. He extends the displayed interval for speed to 0 to 6000 rotations per minute. He also sets the focal point to the non-converged simulation run to ensure that the models compensate for this error. Setting load signal as active dimension generates variation graphs in steps of 10 percent. Despite their comparable error statistics, the shapes of the function graphs differ significantly. Unlike "Torque4a" (solid graphs), the model "Torque4b" (dashed graphs) turns out to be highly sensitive to changes in load signal at high speeds, which is considered undesirable. Even worse, "Torque4b" predicts significant decreases in torque for increasing load signal at low speeds, which is implausible regardless of the values of other model parameters (like pOIM). As a result, "Torque4a" is the better choice.

7.5.2 User Feedback

Based on their own experiences and on the feedback gained from customer training, the overall feedback of our interviewees was very positive. Their former workflow for model identification involved an in-house tool and standard software in engineering like Matlab [172]. Compared to these software packages, our interviewees appreciated that HyperMoVal supports analyzing particular questions and changing scenarios instantaneously, e.g., skimming through the model behavior for different validation points. In their experience, standard software like Matlab requires a setup-time of several minutes for writing and modifying some lines of code for each step of an analysis, and they provide limited interactivity for their visualizations. As a result, HyperMoVal and its integration with other views allows engineers to validate models in detail on average one magnitude faster. As an application expert stated, "*the design [of HyperMoVal] and the high interactivity encourage the analysis of reasons for mismatches. It also facilitates comparisons between models which would otherwise have been omitted in many cases due to the involved effort. Being able to validate a model in detail before usage significantly increases the quality and the confidence in the entire process.*"

An objection was that most engineers are still not used to the concept of interaction. While the improvements in speed (e.g., when rotating a 3D surface) have been noticed immediately, most users tend to map their old workflow to the new tool. This particularly applies to the concept of linking different views as discussed in section 7.3, which may require several days for familiarization. On the other hand, features which are common in other engineering applications are also expected in new approaches and thus required to gain acceptance. For HyperMoVal, this includes the support for 3D plots, parity plots, and the ability to change the transfer function to a rainbow color map, which are still popular with engineers despite their known flaws [26]. Perhaps the most challenging aspect of the design process was thus to find a good balance between preserving familiar methods and introducing techniques that are powerful yet unfamiliar to engineers (e.g., linked views). Nevertheless, our interviewees confirmed that all features of HyperMoVal are useful for specific tasks as discussed in former

sections.

Our approach is distributed under the name IMPRESS xD as part of the software suite of AVL List GmbH, a company providing tools for the development of powertrain systems. While it is potentially available to several thousand users, the application of surrogate models is at the beginning and reliable evidence about adoption rates is not yet available. First estimates assume an application by 5 to 10 companies.

7.6 Discussion and Future Work

The validation of regression models comprises *three levels of detail* as reflected in the application scenario of section 7.5.1: (1) Statistical summaries of the entire validation data provide a coarse yet compact information about the global prediction quality. (2) Derived attributes like residuals describe the quality as one value per data point. They enable the identification of local characteristics like badly fitted regions, but are often insufficient to explain such characteristics. (3) A combined visualization of validation data and function graphs of the model provides most information for a certain point of the parameter space. The shape and the gradient of the function offer detailed reasons yet with very local scope.

HyperMoVal is designed to support domain experts at this latter level, while the integrated workflow described in section 7.3 serves the first two levels. Stand-alone implementations of HyperMoVal may decide to additionally cover the levels 1 and 2 by integrating quantitative model characteristics like the statistics of Fig. 7.9e or additional options for coloring, e.g., mapping residuals to color as in Fig. 7.8b. While respective extensions would be straightforward, our implementation employs linked views for these tasks.

Concerning *scalability*, HyperMoVal has been tested with several thousand validation data points. Multi-threading and the use of graphics hardware ensure immediate visual feedback. Interaction concepts like narrowing the relevant region support the perceptual scalability. HyperMoVal has also been tested for models with more than ten parameters, although such surrogate models are rare in practice. In this case, hiding some dimensions is usually tolerable and provides more space for the rest. An upper limit with respect to the number of compared models is four due to the difficult distinction of function graphs and a significant cluttering.

We believe that HyperMoVal is a good example for combining computation and interactive visualization to support a complex task, which is a main issue of visual analytics. The optional integration to multiple linked views fosters a tight loop of computation-based training and visualization-based validation of regression models. Splitting the data into training and validation data is an important aspect in practice [234] and is supported using selections.

However, further steps towards model identification are a key aspect of *future work*. In particular the identification of good training parameters is an optimization problem on its own. Currently, users with little background in machine learning (as are most designers of car engines) need to manually try many combinations of parameters like the cost value or the gamma value of the SVR. This is neither efficient nor does it ultimately guarantee an optimal set of training parameters. Semi-automatic approaches could integrate optimization techniques to quickly identify promising training parameters. Other plans for future work include a long-term field study of the adoption by different groups of engineers, as well as further navigation concepts for the focal point. Finally, we intend to evaluate our approach within other application areas to assess its general applicability.

7.7 Conclusion

This chapter introduced HyperMoVal as interactive visualization to support various tasks during the validation of regression models. The combined visualization of the n-dimensional model and the validation data provides a direct comparison of known and predicted results and it enables to analyze regions with a bad fit in detail. The simultaneous analysis of families of graphs for multiple models helps to assess and compare the physical plausibility of the function behavior. The matrix-based layout of 2D and 3D slices scales to high-dimensional functions, which can easily be navigated using different interaction techniques. Providing model-related attributes like residuals as derived dimensions optionally complements the analysis in other multivariate views. Linking these views via ad-hoc queries may not only be used to filter implausible validation data; it also supports a user-defined selection of training data as a first step towards a workflow for model building that tightly integrates domain knowledge. User feedback suggests that our approach significantly accelerates the identification of high-quality surrogate models for simulating engine physics in real-time. Application experts consider it an important technique for increasing the confidence in the entire process of car engine design. Motivated by these results, we believe that HyperMoVal may also be beneficial to numerous other application areas in science and engineering.

Chapter 8

Conclusions

The main motivation of visual analysis is to turn the information overload of the 21st century into an opportunity. This explains scalability as a core issue of visual analysis. While scalability is a multifaceted topic, the sheer size of data is a key concern that involves both visual and computational challenges. This thesis made a broad range of contributions to enhance computational scalability, to improve the visual scalability of selected visualization approaches and tasks, and to support an analysis of high-dimensional data.

Concerning computational scalability, this thesis contributed a generic architecture that intends to facilitate the use of multi-threading in the development of highly interactive visual analysis tools. A quantitative evaluation demonstrated that the architecture scales with respect to the size of the data and the number of views while providing permanent visual feedback even during continuous interaction. Instantiations in several visual analysis systems and tools show the applicability to many types of visualizations regardless of a particular platform, programming language or graphics API. As a key aspect of computational scalability, communication and synchronization aspects of the architecture have been covered in detail.

Unlike computational scalability, visual scalability directly depends on the employed metaphor and task. This thesis contributed two variants of scatterplots to address visual scalability for different types of data and tasks. For continuous data, a combination of 2D and 3D scatterplots intends to combine the advantages of 2D interaction and 3D visualization. Several extensions improve the depth perception in 3D and address the problem of unrecognizable point densities in both 2D and 3D. As a result, 3D scatterplots have proven advantageous especially when dealing with inherently spatial data like three-dimensional gradients.

For partly categorical data in general and data cubes in particular, the thesis contributed Hierarchical Difference Scatterplots (HDS). The main goal of HDS is to support analytical tasks involving comparisons of categories of different hierarchy levels. Local drill down and several focus+context approaches ensure the scalability to data cubes of any size. As a key benefit with regards to visual scalability, HDS display the difference between categories explicitly in a single visualization, which makes comparisons more intuitive and more precise than relying on multiple views.

Nevertheless, comparisons in HDS – as for most approaches of information visualization – are only qualitative and limited with respect to the involved dimensionality. For this reason, this thesis also contributed an approach for quantifying subsets of the data by means of univariate and bivariate statistical moments for a potentially large number of dimensions.The

approach has proven useful during multiple stages of an analysis. A quick overview over high-dimensional datasets provides initial guidance, while a quantitative comparison of local features like clusters are beneficial in later stages of an analysis.

Striving for scalability to multi-dimensional regression models, dimensionality has also been a major issue in the design of HyperMoVal. The combined visualization of multiple n-dimensional regression models and respective validation data enables a detailed analysis of regions with a bad fit. The integration with other multivariate views allows for a validation in multiple degrees of detail. An interactive selection of training data is a step towards a user-centric workflow for model building.

Apart from issues that are directly related to scalability with respect to large data, this thesis described several applications of visual analysis. Especially HyperMoVal was the result of a close collaboration with application experts in the field of engine design. It was an interesting lesson to see how it took a significant amount of time to develop a mutual understanding of challenges and possibilities. As another important lesson, finding a good balance between preserving familiar methods and introducing powerful yet unfamiliar techniques is challenging but necessary to gain acceptance in a certain target community. However, as the result of the collaboration, engineers considered HyperMoVal an important technique for increasing both efficiency and confidence in the entire process of car engine design.

Such feedback suggests a high impact of the contributions of this thesis also outside the visualization research community. In this context, it is an important aspect that most contributions have been combined in a common software framework for visual analysis called Visplore. The application of model-building (section 7.5.1) illustrates the usefulness of combining multiple contributions of this thesis for complex real-world problems. Moreover, Visplore is mature in order to be commercially distributed to thousands of customers worldwide by the company AVL List GmbH. Besides the scientific contributions, an indirect contribution of this thesis is thus to raise the awareness and to promote the use of visual analysis as such in different application domains.

However, considering the myriad of challenges related to large data, it is obvious that many open problems remain. A major issue concerns the magnitude of what constitutes "large data". The contributions of this thesis have been evaluated with at most a few million data items. While this exceeds the capabilities of most interactive visualizations today, it is still magnitudes smaller than what many fields in science or business encounter on a daily basis. Moreover, the contributions have only been evaluated with static datasets. Streaming data, for example, involves additional complexity.

In general, dealing with scalability often involves trade-offs. Making use of parallel hardware, for example, typically increases not only the computational scalability, but also the complexity of writing and testing respective software, which explains the need for the proposed multi-threaded architecture. Advanced interaction concepts like multiple linked views enable to analyze complex data, but also require more time for familiarization, as learned from the design of HyperMoVal. As a consequence of such trade-offs, many scalability issues can only be solved in the context of concrete applications and tasks. For all these reasons, scalability will remain a core issue of visual analysis and will motivate as much research in the future as it has done in the past.

Acknowledgments

First of all, I would like to thank my advisor *Prof. Eduard Gröller*, who provided important input and help during the supervision of my work and who repeatedly encouraged me to continue and finish this thesis. Prof. Eduard Gröller always cordially agreed to take enough time when I needed him for fruitful discussions and friendly advice.

A very special thank-you also goes to *Prof. Helwig Hauser* from the University of Bergen, Norway. Prof. Hauser introduced me to visual analysis and aroused my fascination for this topic. His visionary ideas have been a major source of inspiration for my work during the last years, and I would like to thank him for finally accepting the position of reviewing my PhD work.

Furthermore, there are many people who contributed in one way or the other to this thesis. Most importantly, I thank my colleague *Wolfgang Berger*. Wolfgang did not only implement the approaches of the chapters 6 and 7, but he was also involved in preparing the respective publications and he was always willing to discuss ideas with me. I also thank my colleagues *Matthias Buchetics* and *Martin Brunnhuber*, who implemented the approach of chapter 5. Other people who supported the work as co-authors of publications are (in alphabetical order) *Robert Kosara*, *Philipp Muigg*, and *Christian Tominski*. I also want to thank *Helmut Doleisch*, *Kresimir Matkovic*, and *Prof. Heidrun Schumann* for their constructive comments.

Most of this work has been done in the *VRVis Research Center*, which is funded in part by the Austrian Funding Agency *FFG*. Special thanks go to the CEO of VRVis, *Georg Stonawski*, who has always supported my work and my ambitions. Furthermore, much of this work would not have been possible without the VRVis company partner *AVL List GmbH*. AVL not only provided some of the datasets used throughout this thesis, but actively stimulated research by sharing their needs and by participating in the evaluation. In this context, special thanks go to *Jürgen Krasser* who has been promoting visual analysis within AVL for many years. I also thank *Johann Wurzenberger*, *Ivo Prah*, *Natalija Galovic*, and *Florian Spendlingwimmer* for their support of the evaluation.

The most important thank-you goes to my family. I am deeply indebted to my parents *Christine* and *Gerhard*, who have been supporting my interest in computers since I was a child and who have always believed in me. A particularly special and warm thank-you goes to my wife *Caroline* who has always encouraged and motivated me to pursue my research and to finish this thesis. Caro, thank you so much for tolerating the many long and lonely nights when I was working on this thesis and thanks for not complaining about my bad mood during much of the write-up! Finally, I hug my son *Christian*, who has not seen his daddy as often as he would have deserved to, your daddy loves you!

Curriculum Vitae

About Harald Piringer:

Dipl.-Ing. Harald Piringer, born on 28^{th} November 1978, in Vienna, Austria, as the first son of Christine Piringer (maiden name Woldan) and Dipl.-Ing. Gerhard Piringer,
married with Dipl.-Ing. Caroline Piringer (maiden name Langer),
one son Christian Piringer.

Contact information:

Flötzersteig 284/B4, A-1140 Wien, email: piringer@vrvis.at

Professional Activities:

- From June 1998 to Sept. 1999: **Technical support** for Nextra Telekom GmbH in Vienna, Austria
- From March 2000 to Sept. 2000: **Web-development** for Gentics EDV Dienstleistungen GmbH in Vienna, Austria
- From Oct. 2001 to Feb. 2003: **Design and development of entertainment software** for Dion Software GmbH in Vienna, Austria
- Since Aug. 2003: **Researcher** at the VRVis Research Center in Vienna, Austria
- Since Oct. 2007: **Project Leader** at the VRVis Research Center in Vienna, Austria
- Since Jan. 2010: **Head of Group "Visual Analysis"** at the VRVis Research Center in Vienna, Austria

Activities Related to Scientific Work:

- **2 computer science projects** during my studies of computer science at the Institute of Computer Engineering, Vienna University of Technology, about Real-time Systems, Prof. Hermann Kopetz (Oct. 2000 – Nov. 2001)
- My **diploma thesis** *"Occlusion Culling Using Hardware-Based Occlusion Queries"* at the Institute of Computer Graphics, Vienna University of Technology (Aug. 2002 – May 2003)
- **Reviewer for the conferences** *IEEE Information Visualization* (2006 – 2011), the *Eurographics / IEEE Symposium on Visualization* (2008 – 2010)

Activities Related to Teaching:

- Coach of more than 15 **student projects** and **seminar works** in the studies of computer science, including those of Jürgen Platzer, Murat Sari, Johannes Kehrer, Andreas Ammer, Matthias Buchetics, Matej Novotny, and others
- Coach and supervisor of the **diploma theses** of Matthias Buchetics, Wolfgang Berger, Matthias Froschauer, Roland Boubela, and Stephan Pajer.
- Involved in the Organization of the course on **Information Visualization** at the Institute of Computer Graphics, Vienna University of Technology, in 2005
- Lecturing **Information Visualization** at the Fachhochschule Technikum Wien, since 2007.

Education:

- **Volksschule** (primary school) in Klosterneuburg, Austria (Sept. 1985 – June 1989)
- **BG/BRG Klosterneuburg** (combined secondary school and high school) in Klosterneuburg, Austria (Sept. 1989 – June 1997). **Graduation** (Matura) **with highest distinction**
- **Grundwehrdienst** (military service) at the Magdeburgkaserne Klosterneuburg, Austria (Oct. 1997 – May 1998).
- **Studies of Informatik (computer science)** at the Vienna University of Technology, Austria (Oct. 1998 – June 2003)
- Finishing of **diploma thesis** *"Occlusion Culling Using Hardware-Based Occlusion Queries"* at the Institute of Computer Graphics, Vienna University of Technology, in May 2003. **Graduation (with highest distinction) to "Diplom-Ingenieur der Informatik" (computer science)** in June 2003
- Since Sept. 2009: further studies of Informatik (computer science) at the Vienna University of Technology, resulting in the work on this **PhD thesis** at the VRVis Research Center

Reviewed Publications:

Jiri Bittner, Michael Wimmer, Harald Piringer, and Werner Purgathofer
Coherent Hierarchical Culling: Hardware Occlusion Queries Made Useful,
Computer Graphics Forum, 23(3), pp. 615 – 624, 2004.

Harald Piringer, Robert Kosara, and Helwig Hauser
Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots,
Proccedings of the 2nd International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2004), pp. 49 – 60, 2004.

Philipp Muigg, Johannes Kehrer, Steffen Oeltze, Harald Piringer, Helmut Doleisch, Bernhard Preim, and Helwig Hauser
A Four-Level Focus+Context Approach to Interactive Visual Analysis of Temporal Features in Large Scientific Data,
Computer Graphics Forum, 27(3), pp. 775 – 782, 2004.

Harald Piringer, Wolfgang Berger, and Helwig Hauser
Quantifying and Comparing Features in High-Dimensional Datasets,
Proccedings of the 6th International Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2008), pp. 240 – 245, 2008.

Harald Piringer, Christian Tominski, Philipp Muigg, and Wolfgang Berger
A Multi-Threading Architecture to Support Interactive Visual Exploration,
IEEE Transactions on Visualization and Computer Graphics, 15(6), pp. 1113 – 1120, 2009.

Harald Piringer, Matthias Buchetics, Helwig Hauser, and Eduard Gröller
Hierarchical Difference Scatterplots - Interactive Visual Analysis of Data Cubes,
SIGKDD Explorations, 11(2), pp. 49 – 58, 2009.

Harald Piringer, Wolfgang Berger, and Jürgen Krasser
HyperMoVal: Interactive Visual Validation of Regression Models for Real-Time Simulation,
Computer Graphics Forum, 29(3), pp. 983 – 992, 2010.

Wolfgang Berger and Harald Piringer
Peek Brush: A High-Speed Lightweight Ad-Hoc Selection for Multiple Coordinated Views,
Proccedings of the International Conference on Information Visualization (IV 2010), pp. 140 – 145, 2010.

Wolfgang Berger and Harald Piringer
Interactive Visual Analysis of Multiobjective Optimizations,
Proccedings of the IEEE Conference on Visual Analytics Science and Technology (VAST 2010), pp. 215 – 216, 2010.

Wolfgang Berger, Harald Piringer, Peter Filzmoser, and Eduard Gröller
Uncertainty-Aware Exploration of Continuous Parameter Spaces Using Multivariate Prediction,
Accepted for publication in *Computer Graphics Forum*, to appear 2011.

Bibliography

[1] C. Ahlberg. Spotfire: an Information Exploration Environment. *SIGMOD Record (ACM Special Interest Group on Management of Data)*, 25(4):25–29, 1996.

[2] C. Ahlberg and B. Shneiderman. Visual Information Seeking: Tight Coupling of Dynamic Query Filters with Starfield Displays. In *CHI '94: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems*, pages 313–317. ACM, 1994.

[3] 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.

[4] G. Andrienko and N. Andrienko. Spatio-Temporal Aggregation for Visual Analysis of Movements. In *Proc. of the 3rd IEEE Symposium on Visual Analytics Science and Technology (VAST 2008)*, pages 51–58. IEEE Computer Society, 2008.

[5] G. Andrienko, N. Andrienko, S. Bremm, T. Schreck, T. von Landesberger, P. Bak, and D. Keim. Space-in-Time and Time-in-Space Self-Organizing Maps for Exploring Spatiotemporal Patterns. *Computer Graphics Forum*, 29(3):913 – 922, 2010.

[6] G. Andrienko, N. Andrienko, and S. Wrobel. Visual Analytics Tools for Analysis of Movement Data. *SIGKDD Explorations*, 9:38–46, December 2007.

[7] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity Clustering of Dimensions for an Enhanced Visualization of Multidimensional Data. In *Proc. IEEE Symposium on Information Visualization 1998 (InfoVis '98)*, pages 52–60, 1998.

[8] L. Anselin. What is Special about Spatial Data? Alternative Perspectives on Spatial Data Analysis. Technical Report 89-4, National Center for Geographic Information and Analysis, 1989.

[9] A. Asuncion and D. J. Newman. UCI Machine Learning Repository. `http://www.ics.uci.edu/~mlearn/MLRepository.html`, Last visited on Apr. 26th, 2011, January 2008.

[10] BBC Backstage. A Quick Illustrated History of Visualisation. `http://backstage.bbc.co.uk/data_art/resources/history_of_vis.php`, Last visited on Feb. 1st 2011.

[11] M. Baldonado, A. Woodruff, and A. Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 110–119, 2000.

[12] G. Di Battista, P. Eades, R. Tamassia, and I. G. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Prentice Hall PTR, 1998.

[13] D. Baur, F. Seiffert, M. Sedlmair, and S. Boring. The Streams of Our Lives: Visualizing Listening Histories in Context. *IEEE Transactions on Visualization and Computer Graphics*, 16:1119–1128, 2010.

[14] D. B. Beard and J. Q. Walker. Navigational Techniques to Improve the Display of Large Two-Dimensional Spaces. *Behaviour & Information Technology*, 9(6):451–466, 1990.

[15] B. Becker. Volume Rendering for Relational Data. In *Proc. IEEE Symposium on Information Visualization 1997 (InfoVis '97)*, pages 87–91, 1997.

[16] R. Becker and W. Cleveland. Brushing Scatterplots. *Technometrics*, 29(2):127–142, 1987.

[17] B. B. Bederson. Fisheye Menus. In *Proc. of the 13th Annual ACM Symposium on User Interface Software and Technology (UIST '00)*, pages 217–225. ACM, 2000.

[18] B. B. Bederson, A. Clamage, M. P. Czerwinski, and G. G. Robertson. DateLens: a Fisheye Calendar Interface for PDAs. *ACM Transactions on Computer-Human Interaction*, 11:90–119, 2004.

[19] M. Berry and G. Linoff. *Data Mining Techniques, 2nd Edition*. Wiley, 2004.

[20] E. Bertini and D. Lalanne. Investigating and Reflecting on the Integration of Automatic Data Analysis and Visualization in Knowledge Discovery. *SIGKDD Explorations*, 11:9–18, December 2009.

[21] E. Bertini and G. Santucci. Give Chance a Chance: Modeling Density to Enhance Scatter Plot Quality through Random Data Sampling. *Information Visualization*, 5(2):95–110, 2006.

[22] 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 20th annual conference on Computer graphics and interactive techniques*, pages 73 – 80, 1993.

[23] 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.

[24] F. Boecking, U. Dohle, J. Hammer, and S. Kampmann. Passenger Car Common Rail Systems for Future Emission Standards. *Motortechnische Zeitschrift (MTZ)*, 66(7-8):552–557, 2005.

[25] P. A. Boncz. *Monet, a Next-Generation DBMS Kernel for Query-Intensive Applications*. PhD thesis, CWI Amsterdam, 2002.

[26] D. Borland and R. Taylor. Rainbow Color Map (Still) Considered Harmful. *IEEE Computer Graphics and Applications*, 27(2):14–17, 2007.

[27] J. Bottger, M. Balzer, and O. Deussen. Complex Logarithmic Views for Small Details in Large Contexts. *IEEE Transactions on Visualization and Computer Graphics*, 12:845–852, 2006.

[28] N. Boukhelifa, J. C. Roberts, and P. J. Rodgers. A Coordination Model for Exploratory Multi-View Visualization. In *Proc. of the Conf. on Coordinated and Multiple Views In Exploratory Visualization (CMV '03)*, pages 76 – 85. IEEE Computer Society, 2003.

[29] I. Buck, T. Foley, D. Horn, J. Sugerman, K. Fatahalian, M. Houston, and P. Hanrahan. Brook for GPUs: Stream Computing on Graphics Hardware. *ACM Transactions on Graphics*, 23:777–786, 2004.

[30] A. Buja, J. McDonald, J. Michalak, and W. Stuetzle. Interactive Data Visualization Using Focusing and Linking. In *Proceedings IEEE Visualization '91*, pages 156–163, 1991.

[31] T. Büring, J. Gerken, and H. Reiterer. User Interaction with Scatterplots on Small Screens – A Comparative Evaluation of Geometric-Semantic Zoom and Fisheye Distortion. *IEEE Transactions on Visualization and Computer Graphics*, 12:829–836, 2006.

[32] T. Butkiewicz, W. Dou, Z. Wartell, W. Ribarsky, and R. Chang. Multi-Focused Geospatial Analysis Using Probes. *IEEE Transactions on Visualization and Computer Graphics*, 14:1165–1172, 2008.

[33] S. Callahan, L. Bavoil, V. Pascucci, and C. Silva. Progressive Volume Rendering of Large Unstructured Grids. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):1307–1314, 2006.

[34] S. Card, J. MacKinlay, and B. Shneiderman. *Readings in Information Visualization: Using Vision to Think*. Morgan Kaufmann Publishers, 1998.

[35] A. Cedilnik, B. Geveci, K. Moreland, J. Ahrens, and J. Favre. Remote Large Data Visualization in the ParaView Framework. In *6th Eurographics Symp. on Parallel Graphics and Visualization*, pages 163–170, 2006.

[36] S. Chan, L. Xiao, J. Gerth, and P. Hanrahan. Maintaining Interactivity While Exploring Massive Time Series. In *Proc. of the 3rd IEEE Symposium on Visual Analytics Science and Technology (VAST 2008)*, pages 59 – 66. IEEE Computer Society, 2008.

[37] C. Chang and C. Lin. LIB-SVM. `http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, Last visited on Feb. 17th 2010.

[38] R. Chang, G. Wessel, R. Kosara, E. Sauda, and W. Ribarsky. Legible Cities: Focus-Dependent Multi-Resolution Visualization of Urban Relationships. *IEEE Transactions on Visualization and Computer Graphics*, 13:1169–1175, 2007.

[39] S. Chaudhuri and U. Dayal. An Overview of Data Warehousing and OLAP Technology. *SIGMOD Record*, 26:65–74, 1997.

[40] J. Chen, A. M. MacEachren, and D. J. Peuquet. Constructing Overview + Detail Dendrogram-Matrix Views. *IEEE Transactions on Visualization and Computer Graphics*, 15:889–896, 2009.

[41] E. Chi. A Taxonomy of Visualization Techniques Using the Data State Reference Model. In *Proc. IEEE Symposium on Information Visualization 2000 (InfoVis 2000)*, pages 69–75. IEEE Computer Society, 2000.

[42] E. Chi, J. A. Konstan, P. Barry, and J. Riedl. A Spreadsheet Approach to Information Visualization. In *ACM Symposium on User Interface Software and Technology*, pages 79–80, 1997.

[43] W. Cleveland. *The Elements of Graphing Data*. Wadsworth Inc, 1985.

[44] A. Cockburn, A. Karlson, and B. B. Bederson. A Review of Overview+Detail, Zooming, and Focus+Context Interfaces. *ACM Computing Surveys*, 41:2:1–2:31, 2009.

[45] E. F. Codd. *Providing OLAP (On-line Analytical Processing) to User-Analysts*. Codd & Date Inc., 1993.

[46] C. Collins and S. Carpendale. VisLink: Revealing Relationships Amongst Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1192 – 1199, 2007.

[47] G. Convertino, J. Chen, B. Yost, Y.-S. Ryu, and C. North. Exploring Context Switching and Cognition in Dual-View Coordinated Visualizations. In *Proc. of the Conf. on Coordinated and Multiple Views In Exploratory Visualization (CMV '03)*, pages 55–62. IEEE Computer Society, 2003.

[48] A. Dasgupta and R. Kosara. Pargnostics: Screen-Space Metrics for Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 16:1017–1026, 2010.

[49] Oxford English Dictionary. *Aggregation*. Oxford University Press, 2009.

[50] Oxford English Dictionary. *Visualization*. Oxford University Press, 2009.

[51] H. Doleisch. SIMVIS: Interactive Visual Analysis of Large and Time-Dependent 3D Simulation Data. In *Winter Simulation Conference*, pages 712–720. WSC, 2007.

[52] H. Doleisch, M. Gasser, and H. Hauser. Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data. In *Proc. of the 5th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2003)*, pages 239–248. Springer-Verlag, 2003.

[53] H. Doleisch and H. Hauser. Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D. In *Journal of WSCG*, volume 10, pages 147–154, Plzen, 2002.

[54] H. Doleisch, M. Mayer, M. Gasser, R. Wanker, and H. Hauser. Case Study: Visual Analysis of Complex, Time-Dependent Simulation Results of a Diesel Exhaust System. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 91–96. Springer-Verlag, 2004.

[55] M. Dörk, D. Gruen, C. Williamson, and S. Carpendale. A Visual Backchannel for Large-Scale Events. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):1129 – 1138, 2010.

[56] D. Dorling, A. Barford, and M. Newman. Worldmapper: The World as You've Never Seen it Before. *IEEE Transactions on Visualization and Computer Graphics*, 12:757–764, 2006.

[57] S. dos Santos and K. Brodlie. Visualizing and Investigating Multidimensional Functions. In *Proc. of the 4th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2002)*, pages 173–ff. Eurographics Association, 2002.

[58] P. Doshi, G. Rosario, E. Rundensteiner, and M. Ward. A Strategy Selection Framework for Adaptive Prefetching in Data Visualization. In *SSDBM '03: Proc. of the 15th Intl. Conf. on Scientific and Statistical Database Management*, pages 107–116. IEEE Computer Society, 2003.

[59] N. R. Draper and H. Smith. *Applied Regression Analysis, 3rd ed.* John Wiley & Sons, 1998.

[60] J. Dykes, A. MacEachren, and M. Kraak. *Exploring Geovisualization*. Elsevier, 2005.

[61] S. Eick and A. Karr. Visual Scalability. *Journal of Computational and Graphical Statistics*, 11(1):22 – 43, 2002.

[62] G. Ellis and A. Dix. By Chance: Enhancing Interaction with Large Data Sets through Statistical Sampling. In *AVI '02: Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 167–176. ACM, 2002.

[63] 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.

[64] G. Ellis and A. Dix. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216 – 1223, 2007.

[65] 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.

[66] N. Elmqvist, D. Thanh-Nghi, H. Goodell, N. Henry, and J.-D. Fekete. ZAME: Interactive Large-Scale Graph Visualization. In *Proc. of the 2008 IEEE Pacific Visualization Symposium (PacificVIS '08)*, pages 215–222. IEEE Pacific, 2008.

[67] G. Faconti and M. Massink. Continuous Interaction with Computers: Issues and Requirements. In *Vol. 3 of the proc. of HCI International 2001*, pages 301–305. Lawrence Erlbaum, 2001.

[68] W. A. Farrand. *Information Display in Interactive Design*. PhD thesis, University of California, Los Angeles, CA, 1973.

[69] S. Feiner and C. Beshers. Worlds within Worlds: Metaphors for Exploring n-Dimensional Virtual Worlds. In *UIST '90: Proc. of the 3rd ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 76–83. ACM, 1990.

[70] J.-D. Fekete. The InfoVis Toolkit. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 167–174. IEEE Computer Society, 2004.

[71] J.-D. Fekete and C. Plaisant. Interactive Information Visualization of a Million Items. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 117 – 124. IEEE Computer Society, 2002.

[72] W. Felger and F. Schröder. The Visualization Input Pipeline – Enabling Semantic Interaction in Scientific Visualization. *Computer Graphics Forum*, 11:139–151, 1992.

[73] D. Feng, L. Kwock, L. Yueh, and R. M. Taylor. Matching Visual Saliency to Confidence in Plots of Uncertain Data. *IEEE Transactions on Visualization and Computer Graphics*, 16:980–989, 2010.

[74] D. Fisher, S. Drucker, R. Fernandez, and S. Ruble. WebCharts: Extending Applications with Web-Authored, Embeddable Visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 16:1157–1163, 2010.

[75] A. Flexer. On the Use of Self-Organizing Maps for Clustering and Visualization. *Intelligent Data Analysis*, 5:373–384, October 2001.

[76] M. Florek and M. Novotny. Interactive Information Visualization Using Graphics Hardware. In *Poster Proceedings of Spring Conference on Computer Graphics*, 2006.

[77] L. Fortnow and S. Homer. A Short History of Computational Complexity. *Bulletin of the EATCS*, 80:95 – 133, 2003.

[78] A. Frederikson, C. North, C. Plaisant, and B. Shneiderman. Temporal, Geographical and Categorical Aggregations Viewed Through Coordinated Displays: a Case Study with Highway Incident Data. In *Proc. Workshop on New Paradigms in Information Visualization and Manipulation*, pages 26 – 34. ACM, NY, 1999.

[79] J. H. Friedman and J. W. Tukey. A Projection Pursuit Algorithm for Exploratory Data Analysis. *IEEE Transactions on Computers*, C-23(9):881–890, 1974.

[80] Y. Frishman and A. Tal. Multi-Level Graph Layout on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 13:1310–1319, 2007.

[81] B. Fry. *Visualizing Data: Exploring and Explaining Data with the Processing Environment*. O'Reilly Media, Inc., 2008.

[82] Y.-H. Fua, M. Ward, and E. Rundensteiner. Hierarchical Parallel Coordinates for Exploration of Large Datasets. In *Proceedings IEEE Visualization '99*, pages 43–50, 1999.

[83] R. Fuchs, J. Waser, and E. Gröller. Visual Human+Machine Learning. *IEEE Transactions on Visualization and Computer Graphics*, 15:1327–1334, 2009.

[84] G. Furnas. The FISHEYE View: A New Look at Structured Files. Technical Memorandum #81-11221-9, Bell Laboratories, Murray Hill, New Jersey 07974, U.S.A., 1981.

[85] G. Furnas. Generalized Fisheye Views. In *Proc. of the ACM CHI '86 Conf. on Human Factors in Computing Systems*, pages 16–23, 1986.

[86] G. Furnas and A. Buja. Prosection Views: Dimensional Inference through Sections and Projections. *Journal of Computational and Graphical Statistics*, 3(4):323–385, 1994.

[87] G. W. Furnas and B. B. Bederson. Space-Scale Diagrams: Understanding Multiscale Interfaces. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '95)*, pages 234–241. ACM Press/Addison-Wesley Publishing Co., 1995.

[88] J. Gantz. The diverse and exploding digital universe. White paper. International Data Corporation, Framingham, MA, 2008.

[89] Gapminder Foundation. Gapminder. `http://www.gapminder.org/`, Last visited on March 12th 2011.

[90] E. W. Gilbert. Pioneer Maps of Health and Disease in England. *Geographical Journal*, 124(2):172–183, 1958.

[91] A.S. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann Publishers, Inc, San Francisco, 1995.

[92] D. E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[93] B. Goldstein. *Sensation and Perception*. Wadsworth, Thomson Learning, 2002.

[94] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.

[95] N. Greene. Environment Mapping and Other Applications of World Projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, November 1986.

[96] G. Grinstein, M. Trutschl, and U. Cvek. High-Dimensional Visualizations. In *Workshop on Visual Data Mining, 7th Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 77–87, 2001.

[97] D. Guo, J. Chen, A. M. MacEachren, and K. Liao. A Visualization System for Space-Time and Multivariate Patterns (VIS-STAMP). *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1461–1474, 2006.

[98] Z. Guo, M. Ward, and E. Rundensteiner. Model Space Visualization for Multivariate Linear Trend Discovery. In *Proc. of the 4th IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 75–82. IEEE Computer Society, 2009.

[99] C. Gutwin and C. Fedak. A Comparison of Fisheye Lenses for Interactive Layout Tasks. In *Proc. of Graphics Interface 2004 (GI '04)*, pages 213–220. Canadian Human-Computer Communications Society, 2004.

[100] C. Gutwin and A. Skopik. Fisheyes are Good for Large Steering Tasks. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '03)*, pages 201–208. ACM, 2003.

[101] C. D. Hansen and C. D. Johnson. *The Visualization Handbook*. Academic Press, 2005.

[102] M. C. Hao, U. Dayal, D. A. Keim, D. Morent, and J. Schneidewind. Intelligent Visual Analytics Queries. In *Proc. of the 2nd IEEE Symposium on Visual Analytics Science and Technology (VAST 2007)*, pages 91–98, 2007.

[103] M. C. Hao, U. Dayal, R. Sharma, D. Keim, and H. Janetzko. Variable Binned Scatter Plots. *Information Visualization*, 9(3):194 – 203, 2010.

[104] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. *SIGMOD Record*, 25:205–216, 1996.

[105] S. Harizopoulos, V. Liang, D. J. Abadi, and S. Madden. Performance Tradeoffs in Read-Optimized Databases. In *Proc. of the 32nd international conference on Very large data bases (VLDB '06)*, pages 487–498. VLDB Endowment, 2006.

[106] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning, 2nd Edition*. Springer, 2009.

[107] H. Hauser. Generalizing Focus+Context Visualization. In *Scientific Visualization: The Visual Extraction of Knowledge from Data*, pages 305 – 327. Springer, 2005.

[108] H. Hauser, F. Ledermann, and H. Doleisch. Angular Brushing of Extended Parallel Coordinates. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 127–130, 2002.

[109] J. Heer and M. Agrawala. Software Design Patterns for Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):853–860, 2006.

[110] J. Heer and M. Agrawala. Design Considerations for Collaborative Visual Analytics. *Information Visualization*, 7:49–62, 2008.

[111] J. Heer and M. Bostock. Declarative Language Design for Interactive Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 16:1149–1156, 2010.

[112] J. Heer, S. Card, and J. Landay. Prefuse: a Toolkit for Interactive Information Visualization. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems 2005 (CHI 2005)*, pages 421–430. ACM, 2005.

[113] J. Heer, J. Mackinlay, C. Stolte, and M. Agrawala. Graphical Histories for Visualization: Supporting Analysis, Communication, and Evaluation. *IEEE Transactions on Visualization and Computer Graphics*, 14:1189–1196, 2008.

[114] Roman Heinzle. *Machine Learning Methods and Their Application to Real-Time Engine Simulation*. PhD thesis, Johannes Kepler University Linz, 2009.

[115] M. Herlihy and N. Shavit. *The Art of Multiprocessor Programming*. Morgan Kaufmann, 2008.

[116] 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.

[117] U. Hinrichs, H. Schmidt, and S. Carpendale. EMDialog: Bringing Information Visualization into the Museum. *IEEE Transactions on Visualization and Computer Graphics*, 14:1181–1188, 2008.

[118] H. Hochheiser and B. Shneiderman. Dynamic Query Tools for Time Series Data Sets: Timebox Widgets for Interactive Exploration. *Information Visualization*, 3:1–18, 2004.

[119] D. Holten. Hierarchical Edge Bundles: Visualization of Adjacency Relations in Hierarchical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):741–748, 2006.

[120] D. Holten and J. van Wijk. Evaluation of Cluster Identification Performance for Different PCP Variants. *Computer Graphics Forum*, 29(3):793 – 802, 2010.

[121] K. Hornbaek and E. Frøkjaer. Reading of Electronic Documents: The Usability of Linear, Fisheye, and Overview+Detail Interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '01)*, pages 293–300. ACM, 2001.

[122] C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: Spreading Aircraft Trajectories Across Views to Support Iterative Queries. *IEEE Transactions on Visualization and Computer Graphics*, 15:1017–1024, 2009.

[123] IBM. Cognos. `http://www.ibm.com/software/data/cognos/`, Last visited on Feb. 3rd 2011.

[124] Advizor Solutions Inc. Advizor. `http://www.advizorsolutions.com`, Last visited on Feb. 3rd 2011.

[125] Kx Inc. kdb+. `http://kx.com/Products/kdb+.php`, Last visited on Feb. 3rd 2011.

[126] S. Ingram, T. Munzner, and M. Olano. Glimmer: Multilevel MDS on the GPU. *IEEE Transactions on Visualization and Computer Graphics*, 15:249–261, 2009.

[127] A. Inselberg and B. Dimsdale. Parallel Coordinates: a Tool for Visualizing Multidimensional Geometry. In *Proceedings IEEE Visualization '90*, pages 361–378, 1990.

[128] V. Interrante and C. Grosch. Strategies for Effectively Visualizing 3D Flow with Volume LIC. In *Proceedings IEEE Visualization '97*, pages 421–424, 1997.

[129] P. Isenberg, D. Fisher, M. Ringel Morris, K. Inkpen, and M. Czerwinski. An Exploratory Study of Co-located Collaborative Visual Analytics around a Tabletop Display. In *Proc. of the IEEE Conference on Visual Analytics Science and Technology (VAST 2010)*, pages 179–186. IEEE Computer Society, 2010.

[130] Y. Ivanov, C. Wren, A. Sorokin, and I. Kaur. Visualizing the History of Living Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 13:1153–1160, 2007.

[131] S. Jayaraman and C. North. A Radial Focus+Context Visualization for Multi-Dimensional Functions. In *Proceedings IEEE Visualization 2002*, pages 443–450. IEEE Computer Society, 2002.

[132] D. H. Jeong, C. Ziemkiewicz, B. Fisher, W. Ribarsky, and R. Chang. iPCA: An Interactive System for PCA-based Visual Analytics. *Computer Graphics Forum*, 28(3):767 – 774, 2009.

[133] D. F. Jerding and J. T. Stasko. The Information Mural: A Technique for Displaying and Navigating Large Information Spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4:257–271, 1998.

[134] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing Structure within Clustered Parallel Coordinates Displays. In *Proc. IEEE Symposium on Information Visualization 2005 (InfoVis 2005)*, pages 125 – 132, 2005.

[135] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing Structure in Visualizations of Dense 2D and 3D Parallel Coordinates. *Information Visualization*, 5:125–136, 2006.

[136] S. Johansson and J. Johansson. Interactive Dimensionality Reduction Through User-defined Combinations of Quality Metrics. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):993–1000, 2009.

[137] C. Johnson, A. Parker, C. Hansen, G. Kindlmann, and Y. Livnat. Interactive Simulation and Visualization. *IEEE Computer*, 32(12):59–65, 1999.

[138] I.T. Jolliffe. *Principle Component Analysis*. Springer-Verlag, New York, 1986.

[139] A. K. Karlson, G. Robertson, D. C. Robbins, M. P. Czerwinski, and G. Smith. FaThumb: A Facet-Based Interface for Mobile Search. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '06)*, pages 711–720. ACM, 2006.

[140] D. Kasik, D. Ebert, G. Lebanon, H. Park, and W. Pottenger. Data Transformations for Computation and Visualization. *Information Visualization*, 8(4):275 – 285, 2009.

[141] T. A. Keahey and E. L. Robertson. Nonlinear Magnification Fields. In *Proc. of the 1997 IEEE Symposium on Information Visualization (InfoVis '97)*, pages 51–58. IEEE Computer Society, 1997.

[142] J. Kehrer, P. Filzmoser, and H. Hauser. Brushing Moments in Interactive Visual Analysis. *Computer Graphics Forum*, 29(3):813 – 822, 2010.

[143] D. Keim. Designing Pixel-Oriented Visualization Techniques: Theory and Applications. *IEEE Transactions on Visualization and Computer Graphics*, 6(1):59–78, 2000.

[144] D. Keim and H. Kriegel. VisDB: Database Exploration using Multidimensional Visualization. *IEEE Computer Graphics and Applications*, 14(5):40–49, 1994.

[145] D. A. Keim. Datenvisualisierung und Data Mining. *Datenbank-Spektrum*, 2(2):30–39, 2002.

[146] D. A. Keim, J. Kohlhammer, G. Ellis, and F. Mansmann, editors. *Mastering The Information Age – Solving Problems with Visual Analytics*. Eurographics, 2010.

[147] D. A. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual Analytics: Scope and Challenges. pages 76–90, Berlin, Heidelberg, 2008. Springer-Verlag.

[148] Daniel A. Keim. Information Visualization and Visual Data Mining. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):1–8, 2002.

[149] R. Kincaid. SignalLens: Focus+Context Applied to Electronic Time Series. *IEEE Transactions on Visualization and Computer Graphics*, 16:900–907, 2010.

[150] A. Kobsa. User Experiments with Tree Visualization Systems. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 9–16. IEEE Computer Society, 2004.

[151] T. Kohonen. *Self Organizing Maps*. Springer Verlag, 1995.

[152] R. Kosara. *Semantic Depth of Field - Using Blur for Focus+Context Visualization*. PhD thesis, Vienna University of Technology, Austria, 2001.

[153] R. Kosara, F. Bendix, and H. Hauser. TimeHistograms for Large, Time-Dependent Data. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 45–54, 2004.

[154] R. Kosara, F. Bendix, and H. Hauser. Parallel Sets: Interactive Exploration and Visual Analysis of Categorical Data. *IEEE Transactions on Visualization and Computer Graphics*, 12:558 – 568, 2006.

[155] R. Kosara, H. Hauser, and D. Gresh. An Interaction View on Information Visualization. In *Eurographics 2003 State-of-the-Art Reports*, pages 123–137, 2003.

[156] R. Kosara, S. Miksch, and H. Hauser. Focus + Context Taken Literally. *IEEE Computer Graphics and Applications*, 22(1):22–29, 2002.

[157] R. Kosara, G. Sahling, and H. Hauser. Linking Scientific and Information Visualization with Interactive 3D Scatterplots. In *Proceedings of the 12th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, pages 133–140, 2004.

[158] M. Kreuseler, N. López, and H. Schumann. A Scalable Framework for Information Visualization. In *Proc. IEEE Symposium on Information Visualization 2000 (InfoVis 2000)*, pages 27–38, 2000.

[159] M. Kreuseler, T. Nocke, and H. Schumann. A History Mechanism for Visual Data Mining. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 49–56. IEEE Computer Society, 2004.

[160] H. Lam, T. Munzner, and R. Kincaid. Overview Use in Multiple Visual Information Resolution Interfaces. *IEEE Transactions on Visualization and Computer Graphics*, 13:1278–1285, 2007.

[161] J. Lamping, R. Rao, and P. Pirolli. A Focus+Context Technique Based on Hyperbolic Geometry for Visualizing Large Hierarchies. In *Proc. of the ACM CHI '95 Conf. on Human Factors in Computing Systems*, pages 401–408, 1995.

[162] C. Law, W. Schroeder, K. Martin, and J. Temkin. A Multi-Threaded Streaming Pipeline Architecture for Large Structured Data Sets. In *Proceedings IEEE Visualization '99*, pages 225–232. IEEE Computer Society, 1999.

[163] E. A. Lee. The Problem with Threads. *IEEE Computer*, 39(5), 2006.

[164] A. E. Lefohn, S. Sengupta, J. Kniss, R. Strzodka, and J. D. Owens. Glift: Generic, Efficient, Random-Access GPU Data Structures. *ACM Transactions on Graphics*, 25:60–99, 2006.

[165] Y. Leung and M. Apperley. A Review and Taxonomy of Distortion-Oriented Presentation Techniques. *ACM Transactions on Computer-Human Interaction*, 1(2):126–160, 1994.

[166] A. Lex, M. Streit, E. Kruijff, and D. Schmalstieg. Caleydo: Design and Evaluation of a Visual Analysis Framework for Gene Expression Data in its Biological Context. In *Proc. of the 2010 IEEE Pacific Visualization Symposium (PacificVIS '10)*, pages 57 – 64, 2010.

[167] 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.

[168] Z. Liu, J. Stasko, and T. Sullivan. SellTrend: Inter-Attribute Visual Analysis of Temporal Transaction Data. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1025 – 1032, 2009.

[169] J. Mackinlay, P. Hanrahan, and C. Stolte. Show Me: Automatic Presentation for Visual Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 13:1137–1144, 2007.

[170] J. Mackinlay, G. Robertson, and S. Card. The Perspective Wall: Detail and Context Smoothly Integrated. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems and Graphics Interface, ACM SIGCHI*, pages 173–176, 1991.

[171] D.L. Massart, J. Smeyers-Verbeke, X. Capron, and K. Schlesier. Visual Presentation of Data by Means of Box Plots. *Practical Data Handling, LCGC Europe*, 18(4):215–218, 2005.

[172] The MathWorks. Matlab. `http://www.mathworks.com/`, Last visited on Feb. 17th 2010.

[173] K. Matkovic, H. Hauser, R. Sainitzer, and E. Gröller. Process Visualization with Levels of Detail. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 67–70. IEEE Computer Society, 2002.

[174] K. Matkovic, M. Jelovic, J. Juric, and Z. Konyha. Interactive Visual Analysis and Exploration of Injection Systems Simulations. In *Proc. of the IEEE Conf. on Visualization 2005*, pages 391–398. IEEE Computer Society, 2005.

[175] T. Mattson, B. Sanders, and B. Massingill. *Patterns for Parallel Programming*. Addison-Wesley Professional, 2004.

[176] P. McCormick and J. Ahrens. *The Visualization Handbook*, chapter Large-Scale Data Visualization and Rendering: A Problem-Driven Approach, pages 533 – 550. Academic Press, 2005.

[177] J. A. McDonald, W. Stuetzle, and A. Buja. Painting Multiple Views of Complex Objects. *SIGPLAN Not.*, 25(10):245–257, 1990.

[178] B. McDonnel and N. Elmqvist. Towards Utilizing GPUs in Information Visualization: A Model and Implementation of Image-Space Operations. *IEEE Transactions on Visualization and Computer Graphics*, 15:1105–1112, 2009.

[179] A. Mead. Review of the Development of Multidimensional Scaling Methods. *The Statistician*, 33:27–35, 1992.

[180] B. S. Michel and H. Zima. SC'08 Workshop: Bridging Multicore's Programmability Gap, 2008.

[181] MicroStrategy. MicroStrategy Reporting Suite. `http://www.microstrategy.com/software/businessintelligence/`, Last visited on Feb. 3rd 2011.

[182] T. Mihalisin, J. Timlin, and J. Schwegler. Visualizing Multivariate Functions, Data, and Distributions. *IEEE Compututer Graphics and Applications*, 11(3):28–35, 1991.

[183] S. Miksch, W. Horn, C. Popow, , and F. Paky. Utilizing Temporal Data Abstraction for Data Validation and Therapy Planning for Artificially Ventilated Newborn Infants. *AI in Medicine*, 8(6):543–576, 1996.

[184] S. Miksch, A. Seyfang, W. Horn, and C. Popow. Abstracting Steady Qualitative Descriptions over Time from Noisy, High-Frequency Data. In *Proc. of the Joint European Conf. on AI in Medicine and Med. Decision Making (AIMDM99)*, pages 281 – 290, 1999.

[185] D. C. Montgomery and G. C. Runger. *Applied Statistics and Probability for Engineers*. Wiley, 2003.

[186] 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 Scientific Data. *Computer Graphics Forum*, 27(3):775–782, 2008.

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

[188] E. Nam, Y. Han, K. Mueller, A. Zelenyuk, and D. Imre. Clustersculptor: A Visual Analytics Tool for High-Dimensional Data. In *Proc. of the 2nd IEEE Symposium on Visual Analytics Science and Technology (VAST 2007)*, pages 75–82. IEEE Computer Society, 2007.

[189] United Nations. UN Millenium Goals. `http://www.un.org/millenniumgoals/`, Last visited on Feb. 1st 2011.

[190] D. Nekrasovski, A. Bodnar, J. McGrenere, F. Guimbretière, and T. Munzner. An Evaluation of Pan & Zoom and Rubber Sheet Navigation with and without an Overview. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI '06)*, pages 11–20. ACM, 2006.

[191] P. Neumann, S. Schlechtweg, and M. Carpendale. ArcTrees: Visualizing Relations in Hierarchical Data. In *Proc. of Eurographics / IEEE VGTC Symposium on Visualization (EuroVis 2005)*, pages 53–60, 2005.

[192] C. North and B. Shneiderman. Snap-Together Visualization: A User Interface for Coordinating Visualizations via Relational Schemata. In *Proc. of the Working Conf. on Advanced Visual Interfaces (AVI '00)*, pages 128–135. ACM, 2000.

[193] B. Nouanesengsy, S. Seok, H. Shen, and V. Vieland. Using Projection and 2D Plots to Visually Reveal Genetic Mechanisms of Complex Human Disorders. In *Proc. of the 4th IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 171–178. IEEE Computer Society, 2009.

[194] M. Novotný and H. Hauser. Outlier-Preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[195] National Academy of Engineering. Grand Challenges For Engineering. `http://www.engineeringchallenges.org/cms/challenges.aspx`, Last visited on Feb. 1st 2011.

[196] G. M. Olson, T. W. Malone, and J. B. Smith. *Coordination Theory and Collaboration Technology*. Lawrence Erlbaum Assoc., 2001.

[197] J. K. O'Regan. Solving the Real Mysteries of Visual Perception: The World as an Outside Memory. *Canadian Journal of Psychology*, 46:461–488, 1992.

[198] J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell. A Survey of General-Purpose Computation on Graphics Hardware. *Computer Graphics Forum*, 26(1):80–113, 2007.

[199] D. A. Patterson and J. L. Hennessy. *Computer Organization and Design: The Hardware/Software Interface*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 3rd edition, 2007.

[200] T. Pattison and M. Phillips. View Coordination Architecture for Information Visualisation. In *Proc. of the 2001 Asia-Pacific Symposium on Information Visualisation - Volume 9 (APVis '01)*, pages 165–169. Australian Computer Society, Inc., 2001.

[201] W. Peng, M. O. Ward, and E. A. Rundensteiner. Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 89–96. IEEE Computer Society, 2004.

[202] C. Plaisant, D. Carr, and B. Shneiderman. Image-Browsers: Taxonomy and Guidelines for Designers. *IEEE Software*, 12:21–32, 1995.

[203] M. Plumlee and C. Ware. Integrating Multiple 3D Views through Frame-of-Reference Interaction. In *Proc. of the Conf. on Coordinated and Multiple Views In Exploratory Visualization (CMV '03)*, pages 34 – 43. IEEE Computer Society, 2003.

[204] M. D. Plumlee and C. Ware. Zooming versus Multiple Window Interfaces: Cognitive Costs of Visual Comparisons. *ACM Transactions on Computer-Human Interaction*, 13:179–209, 2006.

[205] K. Potter, J. Kniss, R. Riesenfeld, and C. Johnson. Visualizing Summary Statistics and Uncertainty. *Computer Graphics Forum*, 29(3):823 – 832, 2010.

[206] International Social Survey Programme. National Identity II. `http://zacat.gesis.org`, Last visited on March 21st 2011, 2003.

[207] R. Rao and S. K. Card. The Table Lens: Merging Graphical and Symbolic Representations in an Interactive Focus + Context Visualization for Tabular Information. In *CHI '94: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 318–322, New York, NY, USA, 1994. ACM Press.

[208] C. K. Reddy, S. Pokharkar, and T. K. Ho. Generating Hypotheses of Trends in High-Dimensional Data Skeletons. In *Proc. of the 3rd IEEE Symposium on Visual Analytics Science and Technology (VAST 2008)*, pages 139 – 146. IEEE Computer Society, 2008.

[209] G. Reina and T. Ertl. Volume Visualization and Visual Queries for Large High-Dimensional Datasets. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pages 255–260, Konstanz, Germany, 2004.

[210] J. C. Roberts. Waltz - An Exploratory Visualization Tool for Volume Data, Using Multiform Abstract Displays. In *Visual Data Exploration and Analysis V, Proc. of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, pages 112–122. SPIE, 1998.

[211] J. C. Roberts. State of the Art: Coordinated & Multiple Views in Exploratory Visualization. In *Proc. of the Fifth International Conf. on Coordinated and Multiple Views in Exploratory Visualization (CMV '07)*, pages 61–71. IEEE Computer Society, 2007.

[212] G. Robertson, D. Ebert, S. Eick, D. A. Keim, and K. Joy. Scale and complexity in visual analytics. *Information Visualization*, 8(4):247–253, 2009.

[213] G. Robertson, R. Fernandez, D. Fisher, B. Lee, and J. Stasko. Effectiveness of Animation in Trend Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1325–1332, 2008.

[214] J. F. Rodrigues, A. Traina, and C. Traina Jr. Frequency Plot and Relevance Plot to Enhance Visual Data Exploration. In *Proc. of the XVI Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pages 117 – 124, 2003.

[215] B. Sajadi, M. Lazarov, M. Gopi, and A. Majumder. Color Seamlessness in Multi-Projector Displays Using Constrained Gamut Morphing. *IEEE Transactions on Visualization and Computer Graphics*, 15:1317–1326, 2009.

[216] B. Sajadi and A. Majumder. Markerless View-Independent Registration of Multiple Distorted Projectors on Extruded Surfaces Using an Uncalibrated Camera. *IEEE Transactions on Visualization and Computer Graphics*, 15:1307–1316, 2009.

[217] R. Sargent. Verification and Validation of Simulation Models. In *WSC '94: Proc. of the 26th Conf. on Winter Simulation*, pages 77–87. Society for Computer Simulation International, 1994.

[218] D. Schmidt, M. Stal, H. Rohnert, and F. Buschmann. *Pattern-Oriented Software Architecture, Volume 2, Patterns for Concurrent and Networked Objects*. John Wiley & Sons, 2000.

[219] J. Scholtz, K. A. Cook, M. A. Whiting, D. Lemon, and H. Greenblatt. Visual Analytics Technology Transition Progress. *Information Visualization*, 8(4):294 – 301, 2009.

[220] T. Schreck, T. von Landesberger, and S. Bremm. Techniques for Precision-Based Visual Analysis of Projected Data. *Information Visualization*, 9(3):181 – 193, 2010.

[221] E. Segel and J. Heer. Narrative Visualization: Telling Stories with Data. *IEEE Transactions on Visualization and Computer Graphics*, 16:1139–1148, 2010.

[222] J. Seo and B. Shneiderman. A Rank-by-Feature Framework for Unsupervised Multidimensional Data Exploration Using Low Dimensional Projections. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 65–72, 2004.

[223] J. Seo and B. Shneiderman. Knowledge Discovery in High-Dimensional Data: Case Studies and a User Survey for the Rank-by-Feature Framework. *IEEE Transactions on Visualization and Computer Graphics*, 12(3):311–322, 2006.

[224] P. Shanbhag, P. Rheingans, and M. des Jardins. Temporal Visualization of Planning Polygons for Efficient Partitioning of Geo-Spatial Data. In *Proc. of Information Visualization (IV'05)*, pages 211 – 218, 2005.

[225] B. Shneiderman. Tree Visualization with Tree-Maps: 2-d Space-Filling Approach. *ACM Trans. Graph.*, 11:92–99, 1992.

[226] B. Shneiderman. Dynamic Queries for Visual Information Seeking. *IEEE Software*, 11(6):70–77, 1994.

[227] B. Shneiderman. The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations. In *Proceedings of the 1996 IEEE Symposium on Visual Languages*, page 336. IEEE Computer Society, 1996.

[228] M. Sifer. User Interfaces for the Exploration of Hierarchical Multi-Dimensional Data. In *Proc. of the 1st IEEE Symposium on Visual Analytics Science and Technology (VAST 2006)*, pages 175–182, 2006.

[229] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.

[230] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting Good Views of High-Dimensional Data Using Class-Consistency. *Computer Graphics Forum*, 28(3):831 – 838, 2009.

[231] A. Slingsby, J. Dykes, and J. Wood. Configuring Hierarchical Layouts to Address Research Questions. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):977–984, 2009.

[232] H. S. Smallman, M. John, H. M. Oonk, and M. B. Cowen. Information Availability in 2D and 3D Displays. *IEEE Compututer Graphics & Applications*, 21:51–57, 2001.

[233] G. Smith, M. Czerwinski, B. Meyers, D. Robbins, G. Robertson, and D. S. Tan. FacetMap: A Scalable Search and Browse Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12:797–804, 2006.

[234] R. Snee. Validation of Regression Models: Methods and Examples. *Technometrics*, 19(4):415–428, November 1977.

[235] Tableau Software. Tableau software. `http://www.tableausoftware.com`, Last visited on March 21st 2011.

[236] R. Spence. *Information Visualization: Design for Interaction (2nd Edition)*. Prentice-Hall, Inc., 2007.

[237] R. Spence and M. Apperley. Data Base Navigation: An Office Environment for the Professional. *Behaviour and Information Technology*, 1(1):43–54, 1982.

[238] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A System for Query, Analysis, and Visualization of Multidimensional Relational Databases. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):52–65, 2002.

[239] C. Stolte, D. Tang, and P. Hanrahan. Query, Analysis, and Visualization of Hierarchically Structured Data Using Polaris. In *KDD '02: Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 112–122, New York, NY, USA, 2002. ACM Press.

[240] C. Stolte, D. Tang, and P. Hanrahan. Multiscale Visualization Using Data Cubes. *IEEE Transactions on Visualization and Computer Graphics*, 9(2):176–187, 2003.

[241] M. C. Stone, K. Fishkin, and E. A. Bier. The Movable Filter as a User Interface Tool. In *Proceedings of the SIGCHI conference on Human factors in computing systems: celebrating interdependence*, CHI '94, pages 306 – 312. ACM, 1994.

[242] M. Stonebraker and U. Cetintemel. One Size Fits All: An Idea Whose Time Has Come and Gone. In *Proc. of the 21st International Conference on Data Engineering (ICDE '05)*, pages 2–11. IEEE Computer Society, 2005.

[243] E. Tanin, R. Beigel, and B. Shneiderman. Incremental Data Structures and Algorithms for Dynamic Query Interfaces. *SIGMOD Rec.*, 25(4):21–24, 1996.

[244] A. Tatu, G. Albuquerque, M. Eisemann, J. Schneidewind, H. Theisel, M. Magnork, and D. Keim. Combining Automated Analysis and Visualization Techniques for Effective Exploration of High-Dimensional Data. In *Proc. of the 4th IEEE Symposium on Visual Analytics Science and Technology (VAST 2009)*, pages 59–66. IEEE Computer Society, 2009.

[245] D. R. Tesone and J. R. Goodall. Balancing Interactive Data Management of Massive Data with Situational Awareness through Smart Aggregation. In *Proc. of the 2nd IEEE Symposium on Visual Analytics Science and Technology (VAST 2007)*, pages 67–74. IEEE Computer Society, 2007.

[246] M. Theus. Interactive Data Visualization using Mondrian. *Journal of Statistical Software*, 7(11):1–9, 11 2002.

[247] J. J. Thomas and K. A. Cook. *Illuminating the Path: The Research and Development Agenda for Visual Analytics*. IEEE Computer Society, 2005.

[248] C. J. Thompson, S. Hahn, and M. Oskin. Using Modern Graphics Architectures for General-Purpose Computing: A Framework and Analysis. In *Proc. of the 35th Annual ACM/IEEE International Symposium on Microarchitecture (MICRO 35)*, pages 306–317. IEEE Computer Society Press, 2002.

[249] W. Tobler. A Computer Movie Simulating Urban Growth in the Detroit Region. *Economic Geography*, 46(2):234 – 240, 1970.

[250] C. Tominski, J. Abello, and H. Schumann. Axes-Based Visualizations with Radial Layouts. In *Proc. of the ACM Symposium on Applied Computing 2004 (SAC'04)*, pages 1242–1247. ACM Press, 2004.

[251] C. Tominski, J. Abello, and H. Schumann. CGV – An Interactive Graph Visualization System. *Computers & Graphics*, 33(9):660–678, 2009.

[252] M. Tory, A. E. Kirkpatrick, S. Atkins, and T. Möller. Visualization Task Performance with 2D, 3D, and Combination Displays. *IEEE Transactions on Visualization and Computer Graphics*, 12:2–13, 2006.

[253] T. Toutin. Qualitative Aspects of Chromo-Stereoscopy for Depth-Perception. *Photogrammetric Engineering & Remote Sensing*, 63(2):193–203, February 1997.

[254] R. C. Tryon and D. E. Bailey. *Cluster Analysis*. McGraw-Hill, New York, 1973.

[255] Y. Tu and H. W. Shen. Balloon Focus: a Seamless Multi-Focus+Context Method for Treemaps. *IEEE Transactions on Visualization and Computer Graphics*, 14:1157–1164, 2008.

[256] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, CT, USA, 1986.

[257] J. W. Tukey. *Exploratory Data Analysis*. Addison-Wesley, 1977.

[258] L. Tweedie, R. Spence, H. Dawkes, and H. Su. Externalising Abstract Mathematical Models. In *CHI '96: Proc. of the SIGCHI conference on Human factors in computing systems*, pages 406–412. ACM, 1996.

[259] F. van Ham, M. Wattenberg, and F. B. Viegas. Mapping Text with Phrase Nets. *IEEE Transactions on Visualization and Computer Graphics*, 15:1169–1176, 2009.

[260] J. van Wijk. The Value of Visualization. In *Proceedings IEEE Visualization 2005*, pages 79 – 86. IEEE Computer Society, 2005.

[261] J. van Wijk and R. van Liere. HyperSlice: Visualization of Scalar Functions of Many Variables. In *Proc. of the 4th Conf. on Visualization*, pages 119–125. IEEE Computer Society, 1993.

[262] J. van Wijk and E. van Selow. Cluster and Calendar Based Visualization of Time Series Data. In *Proc. IEEE Symposium on Information Visualization 1999 (InfoVis '99)*, pages 4 – 9, 1999.

[263] J. J. van Wijk and W. A. Nuij. Smooth and Efficient Zooming and Panning. In *Proc. IEEE Symposium on Information Visualization 2003 (InfoVis 2003)*, pages 15–22. IEEE Computer Society, 2003.

[264] F. B. Viegas, 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*, 13:1121–1128, 2007.

[265] J. S. Vitter. External Memory Algorithms and Data Structures: Dealing with Massive Data. *ACM Computing Surveys*, 33:209–271, 2001.

[266] T. D. Wang, C. Plaisant, B. Shneiderman, N. Spring, D. Roseman, G. Marchand, V. Mukherjee, and M. Smith. Temporal Summaries: Supporting Temporal Categorical Searching, Aggregation and Comparison. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):1049–1056, 2009.

[267] M. O. Ward. A Taxonomy of Glyph Placement Strategies for Multidimensional Data Visualization. *Information Visualization*, 1(3/4):194–210, 2002.

[268] C. Ware. *Information Visualization: Perception for Design, Second Edition*. Morgan Kaufmann, 2004.

[269] M. Wattenberg. Sketching a Graph to Query a Time-Series Database. In *CHI '01 extended abstracts on Human factors in computing systems*, CHI '01, pages 381 – 382, New York, NY, USA, 2001. ACM.

[270] M. Wattenberg. Visual Exploration of Multivariate Graphs. In *Proc. of the SIGCHI Conference on Human Factors in Computing Systems 2006 (CHI 2006)*, pages 811–819. ACM, 2006.

[271] M. Wattenberg and J. Kriss. Designing for Social Data Analysis. *IEEE Transactions on Visualization and Computer Graphics*, 12(4):549 – 557, 2006.

[272] C. E. Weaver. Building Highly-Coordinated Visualizations in Improvise. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 159–166. IEEE Computer Society, 2004.

[273] C. E. Weaver. *Improvise: A User Interface for Interactive Construction of Highly-Coordinated Visualizations*. PhD thesis, University of Wisconsin - Madison, 2006.

[274] C. E. Weaver. Conjunctive Visual Form. *IEEE Transactions on Visualization and Computer Graphics*, 15(6):929–936, 2009.

[275] C. E. Weaver and M. Livny. Improving Visualization Interactivity in Java. In *Proc. of Visual Data Exploration and Analysis*. IS&T/SPIE, 2000.

[276] D. Weiskopf. *GPU-Based Interactive Visualization Techniques*. Springer, 2006.

[277] L. Wilkinson. *The Grammar of Graphics*. Springer-Verlag New York, Inc., 1999.

[278] L. Wilkinson, A. Anand, and R. Grossman. Graph-Theoretic Scagnostics. In *Proc. IEEE Symposium on Information Visualization 2005 (InfoVis 2005)*, pages 21–28, 2005.

[279] B. Wylie and J. Baumes. A Unified Toolkit for Information and Scientific Visualization. In *Proc. of Visualization and Data Analysis (VDA)*, page 72430H. SPIE, 2009.

[280] D. Yang, E. A. Rundensteiner, and M. O. Ward. Analysis Guided Visual Exploration of Multivariate Data. In *Proc. of the 2nd IEEE Symposium on Visual Analytics Science and Technology (VAST 2007)*, pages 83–90, 2007.

[281] J. Yang, A. Patro, S. Huang, N. Mehta, M. O. Ward, and E. A. Rundensteiner. Value and Relation Display for Interactive Exploration of High Dimensional Datasets. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, pages 73–80, 2004.

[282] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive Hierarchical Dimension Ordering, Spacing and Filtering for Exploration of High Dimensional Datasets. In *Proc. IEEE Symposium on Information Visualization 2003 (InfoVis 2003)*, pages 105–112, 2003.

[283] J. Yang, M. O. Ward, and S. Huang. Visual Hierarchical Dimension Reduction for Exploration of High Dimensional Datasets. In *Proc. of the 5th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2003)*, pages 19 – 28, 2003.

[284] J. Yang, M. O. Ward, and E. A. Rundensteiner. InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pages 77 – 84, 2002.

[285] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interactive Hierarchical Displays: a General Framework for Visualization and Exploration of Large Multivariate Data Sets. *Computers & Graphics*, 27(2):265–283, 2003.

[286] J. S. Yi, Y. Kang, J. Stasko, and J. Jacko. Toward a Deeper Understanding of the Role of Interaction in Information Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13:1224–1231, 2007.

[287] B. Yost and C. North. The Perceptual Scalability of Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 12:837–844, 2006.

[288] L. Zadeh. Fuzzy Sets. *Information and Control*, 8:338–353, 1965.

[289] S. Zhao, M. J. McGuffin, and M. H. Chignell. Elastic Hierarchies: Combining Treemaps and Node-Link Diagrams. In *Proc. IEEE Symposium on Information Visualization 2005 (InfoVis 2005)*, pages 57–64. IEEE Computer Society, 2005.

[290] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. Visual Clustering in Parallel Coordinates. *Computer Graphics Forum*, 27(3):1047 – 1054, 2008.