
TECHNISCHE UNIVERSITÄT WIEN

VIENNA UNIVERSITY OF TECHNOLOGY

**MASTERARBEIT**

# Integrating Interactive Visual Analysis of Large Time Series Data into the SimVis System

ausgeführt am VRVis Zentrum für
Virtual Reality und Visualisierung Forschungs-GmbH

unter der Anleitung von
Priv.-Doz. Dipl.-Ing. Dr.techn. Helwig Hauser
und der Mitbetreuung von
Dipl.-Ing. Dr.techn. Helmut Doleisch und
Dipl.-Ing. Philipp Muigg

eingereicht an der Technischen Universität Wien,
Fakultät für Informatik

durch

**Johannes Kehrer**

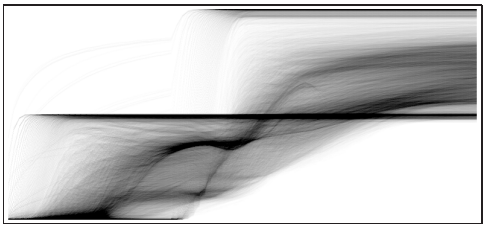
4160 Aigen i.M., Krumauerstraße 7a

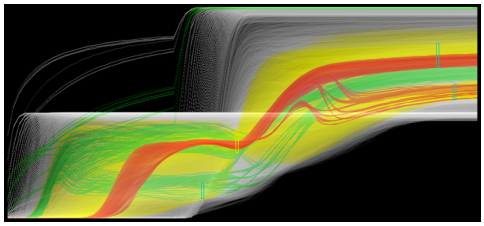
Matrikelnummer: 0526986

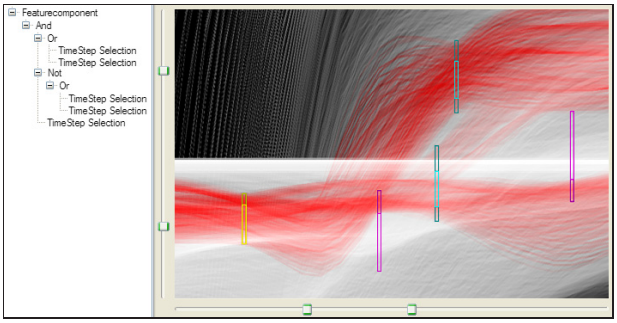
Wien, im Oktober 2007

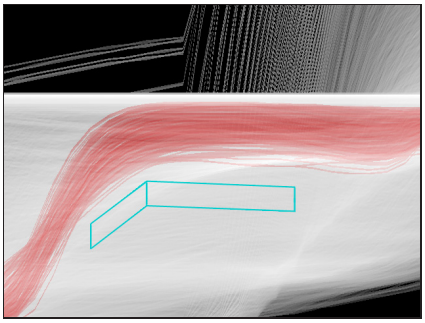
# Integrating Interactive Visual Analysis of Large Time Series Data into the SimVis System

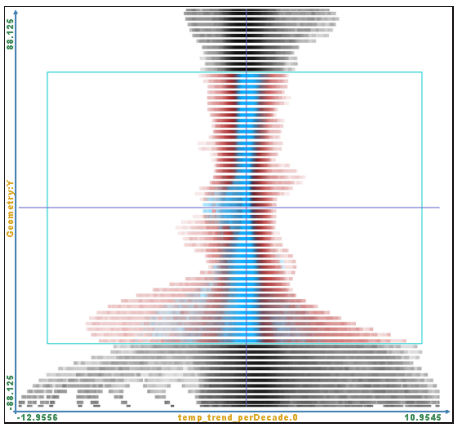
Johannes Kehrer

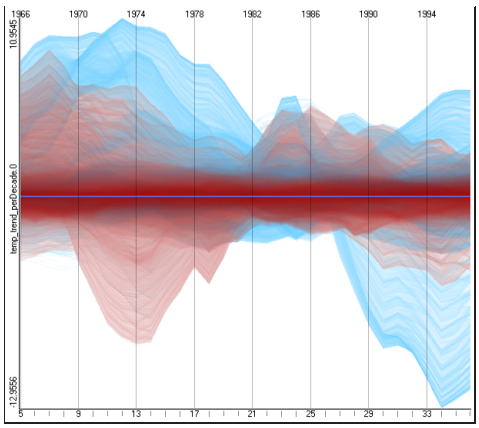









kehrer@VRVis.at

# Abstract

Massive amounts of complex time-dependent information arise in various areas of business, science and engineering. These time series data sets commonly result from the measurement, modeling or the simulation of dynamic processes and contain multiple attributes changing over time. Examples are meteorological data, climate data, financial data, census data, or medical data, to name a few.

In this thesis the *CurveView* for the enhanced interactive visual analysis of multidimensional and *large time series data* is presented. Two approaches are proposed, one for the *interactive visual representation* of the data, and so-called *brushing techniques* allowing the user to select certain interesting subsets of the data (i. e., features) in an intuitive and interactive way. The goals are to enable analysts to gain insight into their data sets, to create, verify or reject hypotheses based on the data, and to explore the temporal evolution of different attributes in order to detect expected structures and to discover unexpected features. The presented solution is integrated into SimVis, a multiple-views system for the visual analysis of time-dependent simulation results.

The data is visualized using *focus+context visualization* techniques: important or selected portions of the data (focus) are visually accented, while the rest of the data (context) is shown in a less prominent style. In doing so, enhanced navigation and orientation is provided to the user. By the application of customizable *transfer functions*, general data trends, visual structures and patterns can be emphasized even within dense regions of the visualization. On the other hand, so-called outliers, which denote time series in low populated areas of the display or important (i. e., brushed) data items hidden in regions of context information, are discriminable in the visualization. By the application of binning techniques large amounts of time-dependent information are transformed into a reduced but still meaningful representation which can be depicted at interactive frame rates.

Furthermore, interactive brushing techniques are provided to the user for analysis purposes. Thus, complex time-dependent features can be specified by applying fuzzy classification to the time series data. Two kinds of brushes exist in the CurveView: *similarity-based brushes* where time series are classified according to their similarity to a user-defined pattern directly sketched in the view; and *time step brushes*, which select time series running through a certain area of the view. In SimVis, the interrelations between the specified features in multiple time-dependent dimensions can be analyzed visually using multiple linked views that show different attributes (i. e., dimensions) of the data.

# Kurzfassung

Riesige Mengen an zeitabhängigen Daten entstehen in den unterschiedlichsten Bereichen von Wirtschaft, Wissenschaft und Technik (z. B. meteorologische Daten, Klimadaten, Aktienkurse, Daten aus der Meinungsforschung oder Medizin). Die entsprechenden Datensätze enthalten so genannten Zeitreihen und resultieren aus der Messung, der Simulation oder der Modellierung von dynamischen Prozessen. Oft sind mehrere Attribute (Dimensionen) enthalten, welche sich über die Zeit verändern.

In dieser Arbeit wird der *CurveView* vorgestellt, eine Möglichkeit zur interaktiven visuellen Analyse von multidimensionalen und großen Datensätzen mit Zeitreihen. Ein spezieller Ansatz zur interaktiven Datenvisualisierung wird präsentiert, außerdem so genannte *Brushing*-Techniken, welche es dem/der BenutzerIn erlauben interessante Merkmale (engl. *features*) direkt am Bildschirm zu selektieren (z. B. mit der Maus). AnalytikerInnen wird es damit erleichtert Einblicke in ihre Datensätze zu erlangen, um neue Hypothesen aufstellen zu können oder bestehende zu verifizieren. Strukturen innerhalb der zeitlichen Entwicklung von unterschiedlichen Datenattributen können visuell erkannt werden, ebenso können unbekannte Merkmale entdeckt werden. Der vorgestellte Ansatz ist in das bestehende *SimVis* System integriert, eine Anwendung zur visuellen Analyse von zeitabhängigen Simulationsdaten, wobei unterschiedliche Darstellungsarten (engl. *views*) zur Verfügen stehen, die untereinander verlinkt sind.

Die Zeitreihen werden mittels *Fokus+Kontext* Visualisierung dargestellt, wobei interessante und wichtige Datenmengen (Fokus) visuell hervorgehoben sind, wohingegen der Rest (Kontext) in abgeschwächter Form dargestellt wird. Diese Vorgehensweise erleichtert es dem/der BenutzerIn durch die Darstellung zu navigieren, ohne dabei die Orientierung zu verlieren. Durch die Verwendung von *Transferfunktionen* können z. B. generelle Datentrends sowie Strukturen und Muster in dichten Bildschirmbereichen verstärkt werden. Außerdem können Sonderfälle hervorgehoben werden – das sind z. B. einzelne Zeitreihen in Bereichen der Visualisierung die nur wenige Daten darstellen oder selektierte Merkmale, die in Regionen mit unselektierten Daten verborgen sind. Durch die Verwendung von speziellen *Binning*-Techniken wird die Datenmengen reduziert, wobei die Bedeutung (Charakteristik) erhalten bleibt – das erleichtert die interaktive Darstellung der zeitabhängigen Information.

Selektionstechniken erlauben die interaktive Analyse der dargestellten Daten. So können komplexe zeitabhängige Merkmale klassifiziert werden, wobei so genannte *fuzzy sets* verwendet werden. Im CurveView stehen Brushes zur Verfügung, die Zeitreihen aufgrund ihrer *Ähnlichkeit* zu einem festgelegten Muster klassifizieren – dieses kann vom Benutzer direkt am Bildschirm als Linienzug gezeichnet werden. Außerdem gibt es so genannte *Time Step Brushes* mit denen Zeitreihen selektiert werden, die durch eine bestimmtes Intervall an einem Zeitschritt laufen. In SimVis können dann die Beziehungen zwischen unterschiedlichen Datenattributen (Dimensionen) – und den dort spezifizierten Merkmalen – in unterschiedlichen verlinkten Views analysiert werden.

# Contents

# 1 Introduction

In our modern society we are confronted with rapidly increasing amounts of complex information. Massive streams of *time-dependent data* arise in various areas of science, business and engineering. They commonly result from the observation (e. g., measurements), modeling, or simulation of dynamic processes. Examples are historical data, meteorological data, climate data, financial data (e. g., stock prices), sensor logs, medical data (e. g., computer tomography (CT) or patient histories), or simulation data such as computational fluid dynamics (CFD), to name a few.

Analysts, engineers, and scientists want to investigate how their information changes over time in order to uncovering interesting and unknown *spatial* and *temporal relationships*, to detect *patterns* of change, major *data trends*, and *outliers* (i. e., anomalies) in their data. Being able to understand time-related developments and changes, allows one "to learn from the past to predict, plan, and build the future" [2]. This can play a major role, when thinking of examples as noticing warning signs (e. g., climate change), the analysis of critical process workflows and developments, forecasts (e. g., project planning or process simulation) and to find and develop alternative scenarios if required.

In this context, the research discipline of *visualization* has proved to be very helpful in order to gain insight into large and complex amounts of (time-oriented) data. Thereby, the information is presented in a visual form to the observer (e. g., in a static image or an interactive application). One takes advantage of the phenomenal capability of the human eye and the brain to process the visual information and to detect interesting visual structures and relationships (e. g., anomalies, patterns, or trends) amongst the depicted data within short time.

The rapid advancement in computer and graphics hardware allows the development of highly interactive *visual analysis systems* that process the information (semi-)automatically and depict it at interactive frame rates. The user is commonly enabled to *zoom* into and *navigate* through the depicted information, furthermore, to specify his/her interest on certain aspects (i. e., smaller subsets) of the data by *selecting* or *querying*. This is then reflected immediately in the visualization, e. g., via highlighting, or filtering. In this context, the integration of proper *interaction* facilities into the analysis and/or visualization system is a key issue.

In section 1.1 an overview on the field of visualization is given. Then, the taxonomy and terms commonly used in this thesis is briefly described (see Sec. 1.2). Finally the problems addressed in this work are described in section 1.3, which also gives an overview of the structure of this thesis.

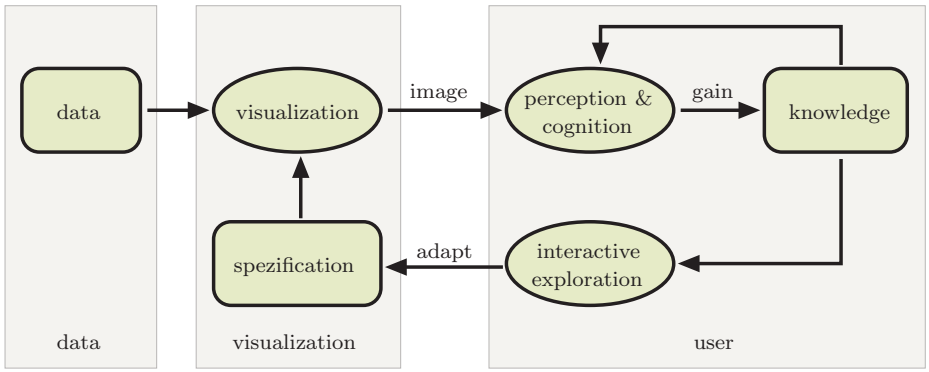



**Figure 1.1:** A simple model of *visualization*, where ellipses denote processes that transform inputs to outputs, and boxes denote containers (adapted from van Wijk [92]).

## 1.1 Visualization

The purpose of *visualization* is to gain *insight* into large and complex amounts of data by presenting the information in a visual form to the observer. In the following, a model is presented giving a simplified view on visualization. Then, further aspects of visualization will be discussed, such as the different application areas of visualization (see Sec. 1.1.2), as well as different visualization purposes (see Sec. 1.1.3).

### 1.1.1 A Simple Model of Visualization

In figure 1.1 a simple and generalized model of visualization (compare to van Wijk [92]) is shown, where boxes denote containers, and ellipses denote processes that transform inputs to outputs. Here, visualization is shown as a *dynamic process*, which transforms *data* into a (time-varying) *visual representation*, such as an image, or the output on the display screen, according to some *specifications*. Thereby, the term data has to be understood in a broader sense (e. g., spatial oriented 3D information, and/or time-dependent information, abstract information, etc.), which will be considered later in Section 1.1.2. The particular specification can include certain (user-defined) parameters affecting the visualization, or issues related to the hardware, as well as the application of different visualization approaches, which can be chosen by the user.

Then, from the users' perspective, *perception* and *cognition* of the resulting data visualization leads to further *knowledge* and (new) insight into the data (see Fig. 1.1). Thereby, the prior knowledge of the user reflects back on the gained knowledge using visualization. This means that an expert in the respective field can extract more information from a visual representation—due to his/her background knowledge—than a normal person. Thereby, the domain expert can gain new insight into certain aspects of the depicted data. On the other hand, someone being new to an area can learn more from the visual representation than an expert. However, the perceptual and cognitive skills of the observer highly influence the amount of gained knowledge.

In the course of the *interactive exploration* process of the data, the user may want to *adapt* the specification (e. g., adapt parameters, zoom, filter) affecting the visualization, based on his/her respective knowledge. This process is an important aspect in visualization, consider for instance the field of *visual analytics* [83, 82] (described later in Sec. 2.1.1), or the seminal paper of Shneiderman [79] introducing the *visual information seeking mantra* ("overview first, zoom and filter, then details-of-demand"). Thereby, the user focuses on further aspects of the data, such as certain *features* (i. e., smaller subsets of the data of special interest), which correspond to some user-defined constraints (e. g., data selection, querying, or filtering). Moreover, he/she may want to zoom and navigate through the data representation, to apply other visualization methods on the data, or to adapt the parameters affecting the visualization, revealing further visual structures (e. g., patterns, trends, or outliers).

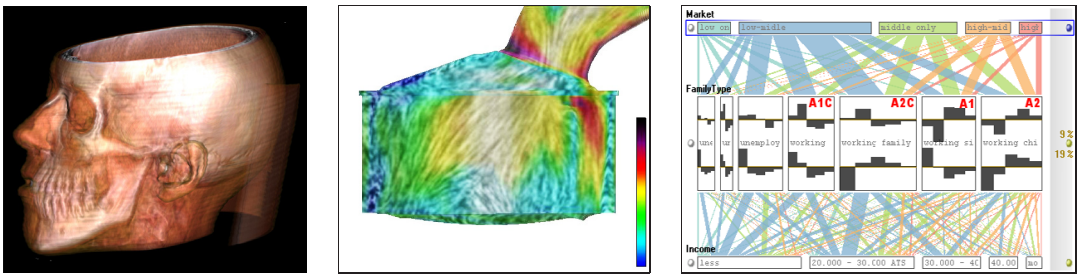
### 1.1.2 Different Application Areas for Visualization

The field of visualization is traditionally classified into two sub-fields, namely the area of *scientific visualization* (SciVis) and *information visualization* (InfoVis). In SciVis, the data has commonly an inherent connection to the spatial domain, i. e., 2D or 3D data, often given with respect to the time domain. In InfoVis, the respective data is more abstract and has commonly no inherent spatial connection (e. g., see Schumann and Müller [78]).

According to Hauser [31], however, this classification is not always very lucky as the terms are misleading themselves, i. e., information is also represented in SciVis, and InfoVis is also scientific. In addition to that, many example exist where the respective characteristics intermix, e. g., where InfoVis techniques are used to visualize data with spatial connection, or InfoVis techniques are incorporated into a SciVis application. Thus, it is important to note that the terms InfoVis and SciVis rather discriminate different visualization communities, than they separate different fields. In the following, several application areas for visualization are described.

**Volume Visualization (VolVis):** In this sub-field of visualization one deals with the exploration and analysis of 3D *volume data*, as they can be found in the medical area (e. g., computer tomography (CT)) but also in the fields of industry or material science. In figure 1.2 (a) a sample visualization of a human head (CT scan) is shown, using *direct volume rendering*. Thereby, a ray of sight is cast into the 3D volume for each pixel in the final image, where the sample points along the ray are composited using a *transfer function*, which maps opacity and color to the samples (this approach is similar to Levoy's [55]).

**Flow Visualization (FlowVis):** In this application area, vector data computed by *flow* simulation (e. g., computational fluid dynamics (CFD)) or measured using appropriate setups is visualized. In figure 1.2 (b) the fuel injection at the surface of a piston cylinder is shown at a certain point in time (see Laramee et al. [54]).




(a) (b) (c)

**Figure 1.2:** Examples for different applications of visualization: (a) *volume visualization* (VolVis) of a human head (CT scan); (b) *flow visualization* (FlowVis) of the fuel injection at the surface of a piston cylinder (image taken from Laramee et al. [54]); (c) *information visualization* (InfoVis) of categorical data using parallel set (image taken from Bendix et al. [6]).

**Information Visualization (InfoVis):** This sub-field of visualization usually deals with the visualization of *abstract* and heterogeneous data, which includes many different dimensions. As the data is commonly not related to the spatial domain it is challenging to represent it in an insightful way to the observer. In this context, the usage of appropriate visual metaphors and interaction techniques is very important. In figure 1.2 (c) a sample information visualization is shown, where a categorical data set is depicted using *parallel sets* (see Bendix et al. [6]).

The above mentioned application areas are kind of standard, however, also others exist, such as the visualization of tensor fields or molecular data visualization.

### 1.1.3 Different Purposes of Visualization

In the following, several purposes of visualization are briefly described (compare to Schumann and Müller [78]). However, it is important to mention, that in nowadays visualization systems these purposes commonly overlap and intermix, since various techniques and aspects are combined.

**Exploration:** This is commonly the first step in data investigation, where *no a-priory information* about the data is know (e. g., no prior knowledge or hypothesis). Thereby, interaction and flexibility of the application are of high importance. The goal is, to discover unknown and unexpected (visual) structures and information within the data (e. g., certain data characteristics, trends, outliers, relations), which lead to insight into the data, and the generation of (new) hypotheses.

**Analysis:** The (emerged or existing) hypothesis and statements on the data, build the basis for visual *data analysis* (see also Sec. 2.1). Here, the goal is to *prove* or *reject* the hypothesis where interaction is a key issue. Thereby, the user commonly selects or queries for certain subsets of the data, which are of high interest (see *brushing*

and *feature specification* in Sec. 2.1.2). In doing so, the visual data representation is affected instantly, i. e., the respective matches are highlighting, and irrelevant information is filtered out, or depicted in a reduced style (see *focus+context visualization* in Sec. 2.1.2). In this thesis an approach for the interactive visual analysis of time-dependent data is proposed.

**Presentation:** In this purpose of visualization, the results gained during the analysis process are visually presented for communication (e. g., static images, animations). Thereby, a high visual quality of the data representation is commonly desirable, to present the findings to some audience (e. g., decision makers, scientists), while interaction is not important. Accordingly, applications focusing on the data analysis are commonly not providing good visual results (e. g., "nice-looking images").

## 1.2 Taxonomy

In the following some basic notations are described and considered, which are used throughout this thesis. According to Müller and Schumann [64], when speaking about data "the term *multidimensional* refers to the dimensionality of the independent variables, while the term *multivariate* refers to the dimensionality of the dependent variables of a data set". In this thesis *time-dependent data* is considered where all variables are given for the independent dimension of time, i. e., multivariate data is given over time. However, as the time-oriented data is also dependent on the spatial position (i. e., each data item is given for a certain 3D position, in relation to time), the term multidimensional is used as well. In section 1.2.1, *time series* data is formally described, and then the *task taxonomy* is presented (see Sec. 1.2.2).

### 1.2.1 Time Series Data

According to the definition of Müller and Schumann [64], a *time series* $\mathcal{D}_i$ represents the evolution of data over time where the time-dependent *data values* $x_{i,j} \in \mathcal{D}_i$ are given at $N$ discrete *time steps* $t_j$ and where $j = 1, \ldots, N$. Consequently, the data elements in the time series are dependent on (i. e., a function of) the respective discrete time step, i. e.,

$$x_{i,j} = \text{data}_i(t_j), \tag{1.1}$$

and form a tuple $(t_j, x_{i,j})$ each. In the context of multi-variate data, the function in equation 1.1 can represent different attributes (e. g., dimensions), therefore, the data items $x_{i,j}$ can also represent different types of data [64]. Moreover, the data values can be given for discrete *points in time*, or for certain *spans of time* having a duration (i. e., time intervals), which will be further discussed in section 2.2.1.

Given a *time-dependent data set* $\mathcal{S}$ consisting of a number of $M$ time series $\mathcal{D}_i$, the set can be expressed as

$$\mathcal{S} = \{\mathcal{D}_i = \{(t_1, x_{i,1}),\ (t_2, x_{i,2}), \ldots, (t_N, x_{i,N})\} \mid i = 1, \ldots, M\}\,. \tag{1.2}$$

The time series are given on a *static sampling or simulation grid*, i. e., the spatial position of the 3D cells where the attributes are measured or simulated does not change over time. Accordingly, the time series $\mathcal{D}_i$ represents the data evolution in the i$^{\text{th}}$ cell of the grid. Equation 1.2 also defines that the data values $x_{i,j}$ of different time series $\mathcal{D}_i$ are given at equal time steps $t_j$ within $\mathcal{S}$. Furthermore, that the data elements belong to the same attribute in the data set and are constrained by the interval $\mathcal{I} = [x_{min}, x_{max}]$, where

$$x_{min} = \min \{x_{i,j} \mid x_{i,j} \in \mathcal{S}\} \quad \text{and} \quad x_{max} = \max \{x_{i,j} \mid x_{i,j} \in \mathcal{S}\}. \tag{1.3}$$

Note, however, that it is not required, that the time series are sampled in regular time intervals, which has to be considered in the proposed approach.

### 1.2.2 Task Taxonomy

Müller and Schumann [64] apply the following *low-level task taxonomy* in their survey on time-oriented data, which was introduced by MacEachren [59]. A good system for the visual analysis of time-oriented information has to answers to the following questions at a glance:

- *Existence of data element:* Does a data element exist at a specific point in time?
- *Temporal location:* When does a certain data element (e. g., pattern) exist in time? Is there any cyclic behavior?
- *Time interval:* How long is the time span from beginning to end of the data element?
- *Temporal texture:* How often does a data element occur?
- *Rate of change:* How fast does a data element change over time, and/or how much difference is there from element to element?
- *Sequence:* In what order do the data elements appear?
- *Synchronization:* Are there data elements, which exist together?

There are also some *high-level tasks* related to the analysis of time-dependent data (compare to Daassi [14]). These tasks are further considered in conjunction with the field of *visual analytics* in chapter 2.

**Navigation:** The user has to be able to search and browse through the depicted time-oriented data by means of interaction (e. g., zooming, panning).

**Observation:** Certain characteristics within a single time series should be visible in the visualization, such as discontinuities, anomalies, or the distribution of the values of a time series. Furthermore, the observer should be able to visually search for patterns (e. g., motifs).

**Comparison:** Here, multiple time series are compared to each other in the visualization. One should be able to search for the *effects* of causality, and to analyze the *correlation*

of different time series to each other. Furthermore to measure the *similarity* between a subsequences in a time series and a reference pattern (this task is addressed in the proposed approach in Sec. 4.3).

**Data Manipulation:** Here, the raw information is transformed into another data representation, e. g., aggregating, or segmenting data values into smaller subsets (see Sec. 2.2.2 and Sec. 2.2.3).

## 1.3 Problem Statement

The interactive visual analysis of time-dependent information is a challenging task, especially when very *large data sets*, containing *multiple dimensions* and several hundred thousands or even millions of data items, are to be visualized and analyzed at interactive frame rates. Thereby, one is commonly interested in the *evolution of the data* over time, to detect interesting patterns, general data trends, data outliers, and anomalies. These goals are addressed in the task taxonomy (e. g., data existence, comparison, observation, etc.), which was briefly described in section 1.2.2.

According to Aigner et al. [2], there are certain **data characteristics**, which have to be considered, when visualizing time-dependent data. The time domain itself has many theoretical aspects, which are addressed in detail in section 2.2.1. Time can, for instance, show *cyclic behavior*, such as the seasons, the changing between day and night, or reoccurring temporal patterns. Furthermore, many *different kinds* of time-related data exist in various research areas, e. g., climate data, financial data, medical data, etc. All of them have their own characteristics, accordingly, many different approaches exist, addressing the certain nature of the data. Moreover, *several attributes* (i. e., dimensions) are commonly given within the data sets. Here, analysts are interested, how these time-oriented attributes are interrelated, e. g., how does the increasing oil price affect business, or how is the $CO_2$ output related to the climate change, etc.

There are also other issues related to the **visual representation** of time-oriented data. Displaying a large number of data items (e. g., several millions) commonly results in overcrowded, and visually cluttered displays, where data items overlap each other. Here, it is very difficult for the observer to identify the interesting visual structures (e. g., patterns and trends) in dense areas. Moreover, the depiction of the massive amount of data commonly takes a few seconds (or minutes), even on nowadays graphics hardware.

### Contribution and Structure of the Thesis

The objective of this thesis is to present the *CurveView* approach for the enhanced **interactive visual analysis** of different kinds of multidimensional **time-dependent data** from various areas (e. g., climate data, meteorological data, simulation data). The approach is seamlessly integrated into **SimVis** [15, 21, and others], a multiple-views system for the visual analysis of complex, time-dependent simulation results and, therefore, applies

the most important concepts. For analysis purpose, SimVis provides *brushing techniques* allowing the user to select certain interesting subsets of the data (*feature specification*) in an intuitive manner, directly in the view. Thereby, fuzzy classification of the data (according to the terminology of fuzzy logic [96]) is applied, called **smooth brushing** [17]. Fractional *degree-of-interest* (DOI) values from the unit interval, which represent the importance of each data item, are altered. Based on these DOI values, the specified features are depicted in multiple linked SimVis views, using **focus+context visualization** [16, 30]. Thereby, important data items (focus) are visually highlighted, while the rest of the data (context) is shown in a reduced style for orientation.

The **CurveView** being part of the SimVis system integrates the above mentioned concepts. For the **interactive visualization** of large time-dependent data sets, the amount of data is reduced—while preserving its characteristics—by the application of *binning techniques* (compare to Novotny and Hauser [67]). The resulting representation can then be visualized in real-time. Moreover, visual clutter reduction is applied using *focus+context visualization*. Using customizable transfer functions, *visual structures* (e. g., general trends, interesting patterns) can be revealed—even within dense regions of the visualization—and *outliers* are preserved.

In addition to the visual approach, the user is able to specify complex **time-dependent features** in an intuitive manner, brushing time series being similar to *user-defined patterns*, which are directly sketched in the CurveView, called . In this context, several types of *similarity-based brushes* were deployed, which also cope with unevenly-spaced time-dependent data. Moreover, time series running through a certain region of the view (i. e., a specified data interval at a time step) can be brushed by the user. The respective features are then depicted instantly in all linked SimVis views using focus+context visualization. Thereby, the interrelations between different data dimensions can be analyzed visually using multiple linked views, which show different aspects (i. e., dimensions) of the data. Based on the visual response of the system, the specified features can be refined in an interactive and iterative brushing process.

---

The thesis is structured as following: In the next chapter the current *state-of-the-art* in the visualization and analysis of time-dependend information is presented. Moreover, the science of visual analytics and the SimVis system are described. The *visualization approach* is presented in chapter 3, and the *brushing techniques* for interactive feature specification are described in chapter 4. Then, some *application examples* are described (see Chap. 5), several details related to the *implementation* and performance of the approaches are presented in chapter 6. Finally, the thesis is *concluded* in chapter 8 where also further working plans are described.

# 2 State of the Art

This chapter gives an overview of the current (2007) state-of-the-art in the visualization and analysis of *time-oriented information* and other work related to this thesis. First, the science of *visual analytics* is presented, which integrates techniques from various research disciplines for enhanced interactive visual analysis of complex data. Then, *SimVis* is presented as a sample application for visual analytics integrating several important concepts (see Sec. 2.1.2). Section 2.2 deals with certain aspects related to the *visualization* (see Sec. 2.2.1) and *analysis* (see Sec. 2.2.2) of time-oriented data. Two recently published works build the basis for section 2.2, namely "*Visualizing Time-Oriented Data: A Systematic View*" [2] and "*Visual Methods for Analyzing Time-Oriented Data*" [1], both done by Aigner et al.

## 2.1 Interactive Visual Analysis

In the introduction of this thesis, the field of visualization was presented in a general manner (see Sec. 1.1). In the current section, the focus is set on the *analytical aspects* of visualization. In this context, the science of *visual analytics* (also known as interactive visual analysis) is presented in section 2.1.1. Then, SimVis is described as an example application of visual analytics (see Sec. 2.1.2), where several important concepts are integrated, which are also related to the approach presented in this thesis.

### 2.1.1 The Interdisciplinary Science of Visual Analytics

Thomas and Cook [83, 82] define *visual analytics* as the interdisciplinary science of analytical reasoning facilitated by interactive, visual and analytical methods. The objective is allowing individuals to gain insight and understanding of rapidly growing and huge amounts of complex data which arise in various areas from business, sciences and engineering. Analysts should be enabled to "detect the expected and discover the unexpected" by interactively and visually examining large datasets. Visual analytics allows them [83, 82]:

- to find interesting patterns, relationships and interconnections within the datasets;
- to draw conclusions and build hypotheses with the examined information;
- to verify or reject these hypotheses; and
- to communicate and present the results of the analytical process in order to make better decisions based on the respective data.

Historically, the science of visual analytics was developed from the field of visualization, which was already described in section 1.1. The basic idea is combining the phenomenal ability of the human mind for understanding complex information visually with automated methods for data analysis and processing. In order to generate, depict and extract information from heterogeneous data sources, visual analytics include sophisticated techniques from various research disciplines, e. g., scientific and information visualization (see Sec. 1.1.2), statistics, cognitive and perceptual science, decision science, knowledge discovery in databases (KDD), human-computer interaction, knowledge representation and many others (compare to Keim et al. [40]).

#### The Knowledge Discovery Process—Keim's Visual Analytics Mantra [40]

Within an iterative process—called the *knowledge discovery process*—analysts gain insight into complex and huge data sets using visualization (see also Sec. 1.1.1). Due to the fact that the raw data is often too complex and large to be represented in a direct manner—the massive amount of data would simply hide the important information—, one has to apply higher levels of data abstraction before visualizing the data. For this purpose, techniques from data preprocessing and automated analysis, knowledge and information representation, interaction, and decision making are included into an iterative data exploration process [40]. This is also true for the visual analysis of time-dependent data, which will be discussed later in section 2.2.2.

The knowledge discovery process in visual analytics can be summed up in Keim's enhancement of Shneiderman's visual information seeking mantra ("overview first, zoom and filter, then details-on-demand", [79]), which is called the *visual analytics mantra* [40]:

> "Analyze First – Show the Important – Zoom, Filter and Analyze Further – Details on Demand".

In the context of this iterative process, *interaction* is a key feature of visual analytics. One takes advantage of human factors such as intuition, creativity, expert and background knowledge, the excellent ability of the eye to detect visual structures and patterns and the flexibility of dealing with unexpected situations. Initially, massive and complex data sets are processed in an automatic analysis step, resulting in a condensed representation containing the important aspects of the data which can be visualized at interactive frame rates. The user gets an overview where he/she can interactively zoom and browse through the representation, select and filter certain data items of special or no interest, respectively. One may also want to trigger further analysis tasks, which provide other aspects and details on demand. These are again visualized, and so on. In the course of this iterative process, the information is revealed step by step, called *information drill down*. Therefore, the user gets deeper and deeper insight into further details of the data.

### 2.1.2 SimVis—an Application for Interactive Visual Analysis

The work presented in this thesis, is integrated into the existing framework of the *SimVis* system [15, 17, 18, 16, 21, and others], which has been developed recently at the VRVis Research Center in Vienna, Austria, and has gained a lot of international scientific acceptance (e. g., winner of the IEEE Visualization Contest 2004 [20], best paper at SimVis 2005 [18], many successful case studies).

SimVis is a multiple-views system for the interactive visual analysis (visual analytics) and exploration of multi-variate, time-dependent 3D data, such as simulation data from the field of engineering [19, 53] or meteorology [20]. Several concepts from visual analytics, which are incorporated in SimVis—and are therefore also relevant for the work in this thesis—, are briefly introduced in the following: *multiple-views* visualization, *focus+context visualization*, *linking and brushing*, and *smooth brushing* for interactive feature specification. However, a general discussion on systems using coordinated and multiple views in exploratory visualization lies beyond the scope of this thesis, hence the interested reader is referred to a current state-of-the-art report by Roberts [74], including many references and example applications.

In figure 2.1 a typical SimVis session is shown, where simulated flow though a diesel particle filter (DPF) is analyzed visually. Features are specified by the user, brushing (selecting) data items directly in a 2D scatterplot (lower left). Here, two components of a multi-dimensional data point are interpreted as coordinates in a Cartesian coordinate system. The selection is refined to hot regions in the histogram view (middle). The 3D view (right) shows the DPF, where the brushed data items are visually highlighted in color using focus+context visualization.

**Focus+Context Visualization:** When visualizing large amounts of information, one is commonly limited by the properties of the output devices, e. g., resolution, available color space or brightness levels. A well-established technique in information visualization to overcome these restrictions is *focus+context visualization* (F+C visualization), which was introduced 1981 by Furnas in his work about *fisheye views* [26]. The basic idea is to give a general overview of the data (*context* part) while—at the same time, within the same view/image—visually highlighting certain subsets of data of special interest for the observer (*focus* part of the data). This interest can be specified, e. g., by brushing the respective data items (see below).

Many enhancements of this technique were developed within the last decades (for an overview, see Hauser [30] or Keim et al. [38]). According to Hauser [30], one can generalize focus+context visualization techniques as the "uneven use of graphics resources (space, opacity, color, etc.) for visualization" which allows the user to visually distinguish between the focus and context portions of the presented data. The details in focus can be inspected while still retaining the roughly represented context data for orientation.

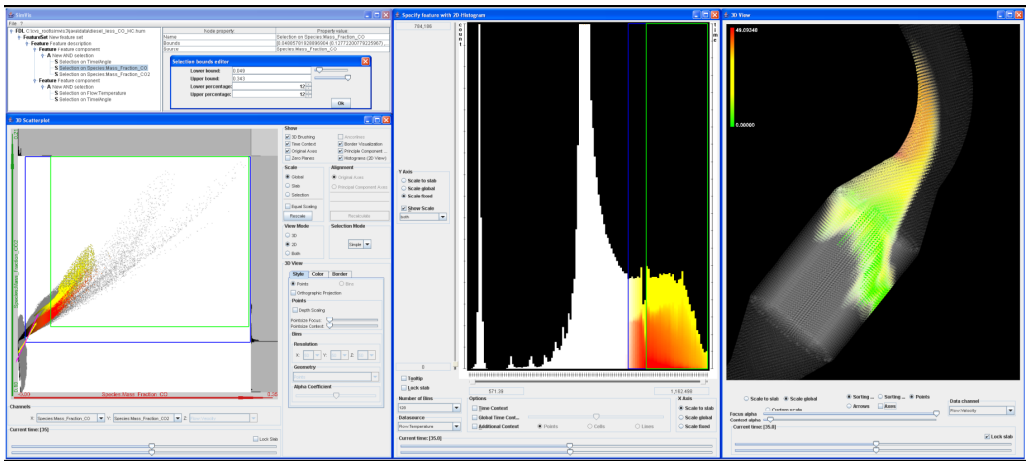


**Figure 2.1:** A sample SimVis scenario: simulated flow through a diesel particle filter (DPF) is visualized—the flow is shown at the time of 35 secs. after the start of the simulation. The user has expressed his/her interest in flow regions of heavy oxidation by interactively brushing data items which exhibit a lot of carbon-oxides in the 2D scatterplot (lower left) and then refining this specification to only apply to hot regions (in the histogram, middle). The 3D view on the right shows a focus+context visualization of the DPF with the brushed data items highlighted in color (color shows velocity magnitudes). Example and image taken from Doleisch [15].

**Multiple-Views Visualization:** In the visual analysis of multi-dimensional data (e.g., with several dozens of dimensions) one is often interested in different aspects of the information, such as different dimensions or attributes: those which are more related to the spatial dependency of the data (e. g., 3D aspects), and those which are more dependent on abstract dimensions or derived attributes of the information, e. g., gradients or velocity. Thus, the SimVis approach combines *attribute views*, such as time-dependent histograms [50], 2D/3D scatterplots [68], parallel coordinates [32]—these are more suitable for feature specification[1], where the user interactively marks data items in a view, called brushing (described later)— and *3D views*. These, in most cases, deal with 3D rendering aspects of the data, e. g., visualization of volume data [62], feature representation using focus+context visualization (described below) and the handling of occlusion.

**Linking and Brushing:** *Brushing* [94, 5, and others] is an important concept in almost every interactive visualization system which enables the user to interactively select a subset of the presented data. Usually this is done by directly marking the data items on the two-dimensional display (e. g., with the mouse). In *linked views* [8, and others], the changes in one view are reflected instantly in all other (linked) views. The combination of multiple linked views and brushing techniques allows the user to interactively explore and analyze the interrelations of the selected information (in different views) and is therefore a well-

[1] *Features* are subsets of data where certain user-defined constraints apply [18, 16].

accepted and useful concept in InfoVis (for an overview see the work of Roberts [73]).

According to Roberts and Wright [75], a *brush* can be specified through a bounding box, a freehand lasso, a line, points or other areas. Various kinds of brushing techniques exist, including standard brushing [5] as well as more advanced approaches, such as *compound brushing* [12] using hierarchical combinations of multiple brushes joined by logical operations, or high dimensional brushes [60], where the brushes are defined in data space rather than in screen space. SimVis uses smooth brushing [17], which is explained below. For an overview of brushing techniques with many references and examples see the work done by Roberts and Wright [75].

**Smooth Brushing for Interactive Feature Specification:** Many visualization and exploration applications in literature only use a binary classification scheme to represent data selections. However, when dealing with simulation data (e. g., computational fluid dynamics (CFD)), it is often not possible to find a certain (clear) threshold which discriminates interesting from uninteresting data (i. e., focus data from the context). Thus, SimVis uses *smooth brushing* (see Doleisch and Hauser [17]) for a fuzzy classification of the data (according to the terminology of *fuzzy logic* [96]).

As described above, the user actively specifies his/her interest on a certain subset of the data displayed in a view via brushing (the associated process is commonly called *feature specification*). In SimVis, this interest is formally represented as a fractional *degree-of-interest* (DOI) value (compare to Furnas [27]) from the unit interval assigned to each data entry. A border region where the DOI values gradually change between data of maximal user interest (DOI value of one, or data in *focus*) and completely uninteresting data (DOI value of zero, or data in *context*) is assumed in smooth brushing. The selection is immediately reflected visually in all (linked) SimVis views through the propagation of the DOI values and by the use of focus+context visualization (see Fig. 2.1).

**The Feature Definition Language (FDL) tree:** SimVis allows the logical combination of multiple brushes using fuzzy-logic operations for complex feature specifications. Thereby, smooth brushes (from multiple views) are organized in a hierarchical *feature definition language* (FDL) tree-like structure, which can be saved to and loaded from XML-files[2]. Amongst others, the following FDL nodes are used in the FDL tree [16]:

**Feature Component:** Each SimVis view with brushing enabled has an associated *feature component* node. All individual brushes (leaves in the FDL tree) in the view are merged using fuzzy set operations (AND, OR, NOT) on the respective DOI values.

**Feature Description:** In a *feature description* node an arbitrary number of feature components can be combined using an implicit-AND operation.

[2]Extensible Markup Language, see http://www.w3.org/XML/

**Feature Set:** Several description nodes can be subsumed in a *feature set* node using an implicit-OR operation. The feature set nodes are located directly below the root node of the FDL tree.

According to the DOI values given at the different levels (nodes) of the FDL tree, the coloring of the data items displayed in an *attribute view* is altered (pure colors for focus data, gray colors for the context). Thereby, *importance-driven color coding* is used for focus+context visualization of the specified features, where relevant colors obliterate (cover) those less important. In a 3D view in SimVis, moreover, the *opacity* of the displayed data items is altered, according to the respective interestingness for the user (DOI information). Thereby, high opaqueness and saturated colors are used to emphasize focus data while the context is depicted as more transparent and uncolored.

## 2.2 The Visualization and Analysis of Time-Dependent Data

According to Keim [40], the visualization and analysis of time-related information is one of the key challenges in *visual analytics* described in section 2.1.1. Huge streams of time-oriented data arise in various areas of business, science and technology, such as financial data, meteorological data, climate data, network monitoring, sensor logs, etc. Analysts commonly want to investigate how this data evolves over time, uncovering *spatial* and *temporal relationships*, detecting interesting and unknown *patterns*, major *data trends*, and *outliers* (anomalies) as well. Being able to understand time-related developments and changes allows one "to learn from the past to predict, plan, and build the future" [2]. This can play a major role in such examples as indicators of climate change, analysis of critical process workflows and developments, forecasts, and the discovery and development of alternative scenarios.

A large number of publications dealing with the visualization and analysis of *time-oriented data* exists. Related surveys on this topic are, for instance, the work done by Silva and Catarci in 2000 [80] giving an overview on techniques for interactive exploration of linear, time-oriented (historical) information. Furthermore, Müller and Schumann [64], who set their focus on *multivariate* time-dependent data, are of importance. In a book written by Andrienko and Andrienko [4] (published 2006), a systematic approach for the exploratory analysis of spatial and temporal data is given, including many examples and related methods, moreover, generic procedures for data exploration are presented.

Two recent publications done by Aigner et al. in 2007 [2, 1] deal with the visualization of time-oriented information, and also build the basis for the aspects discussed in the current section. In the first article [1], the advantages of combining *visual* and *analytical* methods into the interactive visual analysis of time-oriented data are illustrated. The importance of the *user* is pointed out (user-centered visual analysis), e. g., proper interaction methods and using event-based visualization [84]. In the second article of Aigner et al. [2], a *systematic view* on the variety of visualization methods for time-dependent data is given.

A categorization which aims to be helpful for users and researchers to identify future tasks in the visual analytics of time-oriented information is proposed.

The following section deals with certain aspects related to the visualization of time-dependent information. Thereby, the characteristics of the time dimension itself, and issues related to the data are discussed. Section 2.2.2 describes techniques for the interactive visual analysis of time-oriented data. Finally, brushing and querying techniques for the specification of time-dependent features are presented in section 2.2.3.

### 2.2.1 The Visualization of Time-Dependent Data

#### Different Graphical Representations for Time-Oriented Data

Müller and Schumann [64] discriminate between *static* and *dynamic representation*. This discrimination is also crucial in visual analytics, as different tasks and goals which will be discussed in the following are addressed (compare to Aigner et al. [2]). Moreover, *2D* or *3D graphical representation* can also be used in the visualization.

**Static representation vs. dynamic representation:** Initially it has to be noted that the facility of interaction and parametrization in a visualization tool does not influence whether it is considered to be a static or dynamic representation. In this context, *static representation* means still images that are modified only manually by user interaction. In *dynamic representations*, however, the visualization changes automatically over time without the needs of interaction (e. g., slide show or animation). Both are important classes of visualization techniques (see Schumann and Müller [78, Chap. 6.2]) and can therefore also be applied to the visualization of time series data [64, 2]:

**Static representation:** Many visualization techniques known from literature use static representations to display the time-dependent information on a screen while providing proper interaction techniques for exploration. This allows the user to focus on the displayed data, to compare different time spans on the time axis, and to search for patterns and data trends over time. Therefore, one can conclude *quantitative statements* from the observations [64]. On the other hand, these visualization techniques have to cope with know issues when visualizing massive amounts of data, e. g., visual cluttering and overcrowded displays. Advanced techniques have to be deployed to cope with these problems, e. g., data aggregation and segmentation by the use of analysis techniques (these techniques will be discussed later in Sec. 2.2.2 and Sec. 2.2.3).

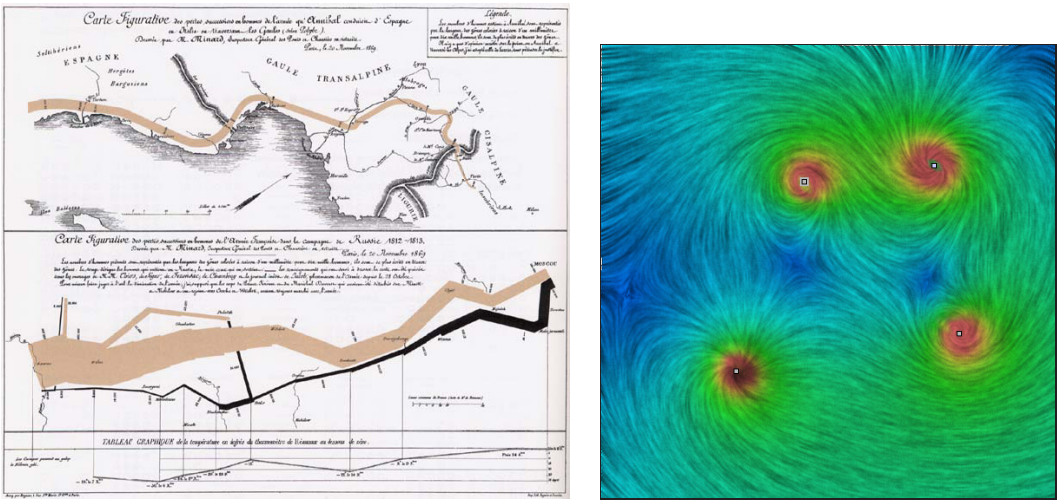
One very early but excellent example of static representation is an image by Charles Joseph Minard, created 1869, which shows Napoleon's army on its march toward Moscow in 1812 (see Fig.2.2 (a)). Tufte considers it as one of the "best statistical graphics ever drawn" [87]. The width of the trails corresponds to the size of the

French army placed over a graphic map, the temperature is displayed below related to dates during the retreat.

**Dynamic representation:** The idea behind dynamic visualization of time-related information is to use the physical dimension of time for representing the data's time dependency. An example is the *animated visualization* of two-dimensional fluid flow shown in Figure 2.2 (b), which is depicted as moving textures using line integral convolution and spot noise (compare to van Wijk [91]). The animation is generated, blending a warped version of the previous image and a number of background images (i. e., white noise images filtered in time and space in order to remove high frequencies).

In the course of an animation, analysts can follow the general development of the displayed data and perceive how it changes over time (e. g., size, color, shape, texture). Therefore, *qualitative statements* can be concluded from the observation [64]. Dynamic representations of time-related data animations can be: faster than the original speed (*time-lapse*); unchanged (*real-time*); and in *slow motion*. However, it is especially difficult to follow long term data changes (e. g., weather) when the animation plays to slowly in relation to the data changes [2].

**2D vs. 3D representation:** The general decision whether to use a 2D or 3D graphical data representation in information visualization is not easy and depends on the specific task (see Schumann and Müller [78, Chap. 6.2]). It is often argued that a 2D representation is sufficient for data analysis as data items can be depicted and interpreted more




(a) (b)

**Figure 2.2:** Static vs. dynamic representation: (a) Napoleon's march on Moscow in 1812 by Charles Joseph Minard (from Tufte [87]); (b) Animated flow visualization (compare to van Wijk [91]).

directly. Furthermore, using a third spatial dimension introduces additional complexity, such as perspective distortion, occlusion, or lost information on back faces. On the other hand, humans experience their environment in three dimensions and are therefore used to perceiving and interpreting 3D information. Additional information can be encoded in the third dimension of the representation, which requires advanced interaction and navigation techniques for exploring the data. Also, some data types inherently require a 3D representation (e. g., 3D flow data [72, 19], volume data [55]).

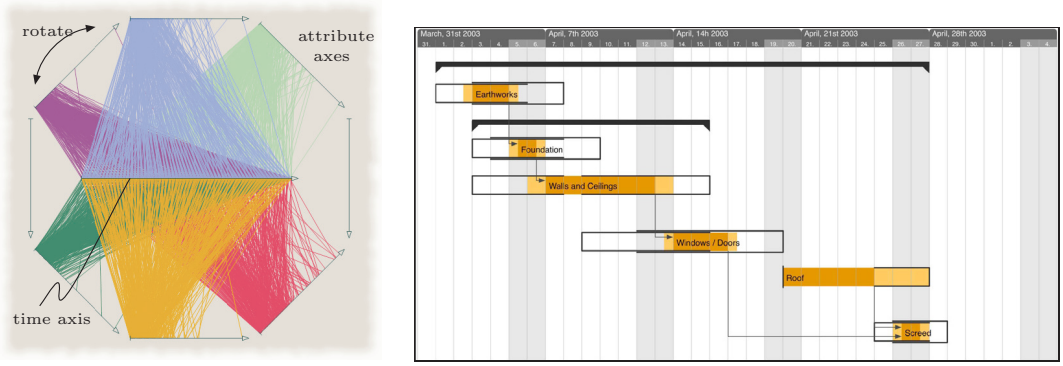
### Several Characteristics of Time

Aigner et al.[2, 1] as well as Müller and Schuman [64] consider certain characteristics of time in their surveys. These are based on a taxonomy done by Frank [25], which will be discussed in the following.

**Temporal primitives: discrete time points vs. time intervals:** A discrete *time point* is considered as a moment in time which has no extent. In contrast, a *time interval* is bounded by a starting and an ending point and has a certain duration. According to Aigner et al. [2], the discrimination of these two *temporal primitives* (time points and time intervals, respectively) is an important aspect in visual analytics. This is due to the fact that data validity is designated by the corresponding temporal primitive. This will be briefly discussed in the following:

**Discrete time points:** When data is given at discrete points on the time axis (e. g., time steps) it can be regarded valid only at the respective instants in time and one does not know how it behaves between adjacent time points. For instance, an account balance does not gradually change amongst two transactions, which should be expressed in the visualization to avoid wrong assessments. Natural phenomenons, like the weather, usually change gradually, therefore it can make sense to interpolate between given time points in the visualization, e. g., by visually connecting the points with a line as it is done in standard *line graphs* (see Harris [29]). In the visualization of time-related data given at discrete time points the focus is usually set on the representation of the data using a rather simple time axis. In most cases, quantitative values (but also qualitative attributes) are displayed, allowing the observer to make quantitative statements based on the data.

The *TimeWheel* [85] is such an example where multivariate time-dependent data is visualized using multiple axes, which are arranged circularly around the time axis (see Fig. 2.3 (a)). Thereby, lines are drawn for each point in time, interconnecting the multivariate data values on the different attribute axes to the respective values on the time axis. Patterns, correlations and trends can be explored best when the respective axes stand parallel to each other. Therefore, the user can rotate the radial arrangement of the axes. Moreover, one can zoom into data ranges of interest on the axes using a slider.




(a) (b)

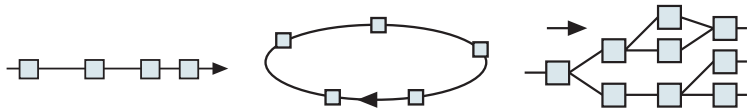
**Figure 2.3:** Time points vs. time intervals: (a) TimeWheel [85] visualizing multivariate time-related data; (b) PlanningLines [3] for the visualization of time intervals, their relations and uncertainties.

**Time intervals:** On the other hand, when data is defined for certain time intervals, an interval scaled time domain is applied (e. g., days, months, years). Examples applications are project planning where different tasks have a certain duration or patient histories in the medical domain. In these cases it makes sense to focus on the representation of the time domain itself, depicting the temporal primitives and the relations among them, while using rather simple data representations.

For instance in *PlanningLines* [3], temporal intervals (e. g., certain activities) and related uncertainties are visualized as shown in Figure 2.3 (b). Each task consists of two encapsulated bars representing the the minimum and maximum duration, which are bounded by two caps depicting the start and end intervals. Thereby, also uncertainties can be visualized, i. e., when planning future activities one often does not know exactly how long a task can take or when it will start or end.

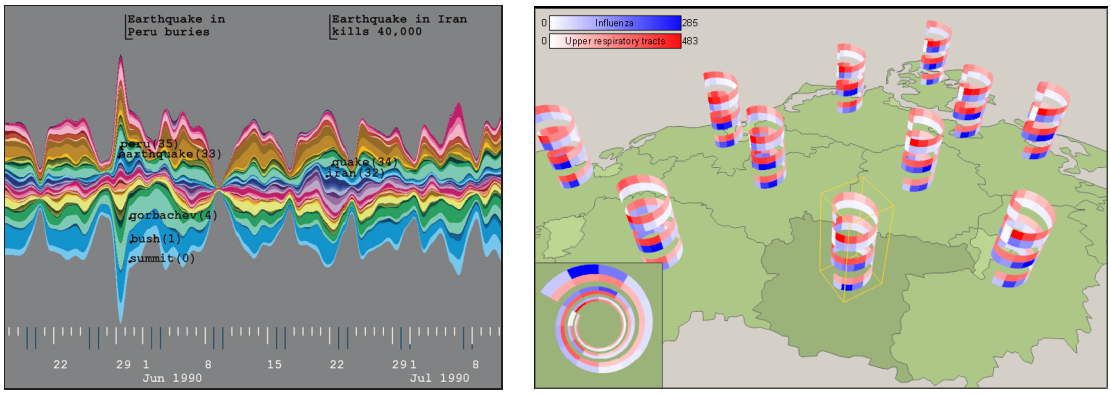
**Structure of time: Linear time vs. cyclic time vs. branching time:** Aigner et al. [2] incorporate the following *structure of time* from Frank's taxonomy [25] into their systematic view on the visualization of time-oriented data (compare also to Müller and Schuman [64]).

*Linear time*, as illustrated in Fiure 2.4 (a), adopts a linear time axis where the temporal primitives are ordered sequentially (but not necessarily evenly spaced) from past to future, which corresponds to the human perception of time. Note that the temporal primitives need not be evenly spaced. Conversely, *cyclic time* (see Fig. 2.4 (b)) is based on the fact that many natural processes follow a cycle (e.g., day and night, seasons of the year). Thereby temporal order—with respect to the cyclic time domain—has no meaning, e. g., sunrise comes before sunset, but sunset also comes before sunrise. Note that a cyclic time axis can also be unrolled to a linear time axis. Figure 2.4 (c) illustrates *branching time* which splits the time axis into alternative scenarios where sequences of actions can happen simultaneously. For this purpose the time axis is represented as a graph.



(a) (b) (c)

**Figure 2.4:** Structures of time: (a) linear time; (b) cyclic time; (c) branching time. Adapted from Aigner et al. [2].




(a) (b)

**Figure 2.5:** Linear time vs. cyclic time: (a) ThemeRiver for analyzing temporal trends of topics in large document collections (image taken from Havre et al. [33]); (b) Helix icons for analyzing cyclic time-dependent patterns of two selected diseases. Using a "tunnel view" reveals hidden information for a selected icon (from Tominski et al. [86]).

**Linear time vs. cyclic time:** A linear time axis is suited very well for the analysis of major data trends and developments over time. The *ThemeRiver* [33] is a related approach, which was originally designed for the visualization of thematic changes in large document collections (see Fig. 2.5 (a)). Thereby, the width of each colored river band represents the strength of a certain theme (i. e., the respective number of occurrences in the media) over time. As the topics are not equally treated (e. g., those near the middle are less distorted), interaction techniques such as re-arrangement are required. The approach is also suitable for other kinds of data (e. g., clustered climate data [65]), which will be considered later in Section 2.2.2.

However, when searching for repetitive (cyclic) *temporal patterns* (e.g., seasonal effects), the use of a cyclic time axis often makes sense. For instance, 3D *helix glyphs* on maps [86] are used to represent the spatio-temporal relations of health data (see Fig. 2.5 (b)). The time-dependent data values are color coded and mapped to the ribbons, which increases in angle and height for each time step in order to create a "tunnel view" (see the left, bottom visualization in Fig. (b)). Thereby, multiple data attributes can be represented by subdividing each ribbon into smaller sub-ribbons. Interaction techniques such as altering the helix' diameter (in order to reveal cyclic behavior) and navigating through the 3D representation are also incorporated.

**Branching time and multiple perspectives:** Branching time is especially useful for decision making, planning applications or the development and evaluation of alternative future scenarios, all of which are essential tasks in visual analytics. For planning it is also required to represent temporal uncertainty (e. g., the task will take 3–4 days) and simultaneous tasks as it was mentioned previously in connection with PlanningLines [3]. In the context of branching time, Frank [25] suggests *time with multiple perspectives* as an additional categorization which enables several (concurrent) points of view on the same event.

### 2.2.2 Visual Analytics of Time-Dependent Data

The visualization approaches for time-dependent data presented in section 2.2.1, are well suited for datasets with several thousand entries. However, when really large amounts of data with, e. g., several hundred thousand or even millions of data entries have to be visually analyzed, the bigger part of these techniques reach their limits resulting in *overdrawing* and *visual clutter*. In addition to that, the usability of the application for the purpose of visual analytics (see Sec. 2.1.1) is highly affected, when the response time to the *interactions* of the user is relatively long (e. g., data selection, drag an item, zoom, pan, etc.), or when updates of the display take longer than a certain time (e. g., 100–200 milliseconds [67]).

According to Keim's *visual analytics mantra* [40], which was briefly described in section 2.1.1, (semi-)automated data reduction and abstraction techniques have to be applied in the visualization process. These techniques transform the time-dependent data into a compressed but still representative form, which enables the observer to better understand the information. Then, this reduced data representation is depicted in real-time, instead of the original (raw) data. The user can explore further details, for instance, by interactively changing the level of data abstraction (e. g., level of details), by zooming, or by browsing through the visualization. Additionally, certain subsets of the data of special interest (features) can be selected by the user (see Sec. 2.2.3) for further analysis or enhanced visualization. Additional information or the original data can be displayed on demand, even though only a small subset of the data (e. g., selected features). These *details on demand* (see Shneiderman [79]) are another important concept in visual analytics (compare to Keim et al. [40]).

In the remainder of this chapter, the focus will be set on two important aspects of **data manipulation** for the purpose of effective visualization and interactive visual analysis of time-oriented data: *data abstraction* is presented in this section and *manual time-dependent feature* and *event specification* (e. g., selection, brushing) is considered later in section 2.2.3.

### Data Abstraction for Effective Visualization

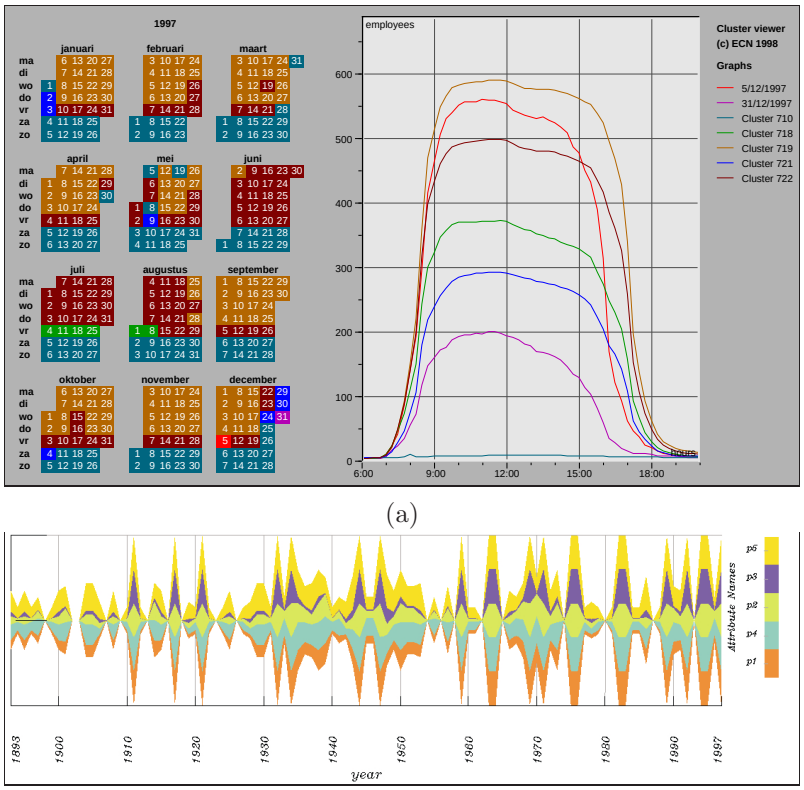
The objective of *data abstraction* is to derive meaningful *qualitative values or patterns* from raw data, which still "convey key ideas while suppressing irrelevant details" [1]. Then the representative form of the data is visualized instead of the original data. Many techniques dealing with the abstraction of temporal data come from the field of *data mining* (see Keogh et al. [44]), such as temporal data classification [71], temporal aggregation [58], motifs and anomaly detection [43, 13], indexing techniques [46] or principle component analysis (PCA) [63], to name just a few. Other approaches transform the data from the time domain into another (abstract) domain, which represents certain data aspects (e. g., trends, frequency) more directly [57]. Examples are Wavelets [11], or spectral techniques such as Discrete Fourier Transform (DFT) [22] or Discrete Cosine Transform (DCT).

Generally speaking, the mentioned data mining techniques are not only useful for *effective* visualization (i. e., to accomplish the desired goals) and *efficient* visualization (i. e., use a minimal amount of resources)—in this context, the reader is referred to the work of Keim et al. [38] on *visual data mining*. Moreover, these techniques can also be used for fast and automated pattern matching and similarity-based comparison of huge time series data. However, a detailed discussion of the data mining approaches lies beyond the scope of this thesis, since this section focuses on the visualization and manual selection of the data. The interested reader is referred to Keogh et al. [44], wherein several data mining techniques are presented and discussed, or the work of Roddick and Spiliopoulou [76] on temporal knowledge discovery. Stacey and McGregor [81] present an overview of temporal abstraction in intelligent clinical data analysis. In the following data aggregation techniques, such as clustering or binning are discussed.

### Temporal Data Aggregation

*Data aggregation* is an important example of data abstraction, where data is summarized in smaller subsets by means of an aggregate function. When data aggregation is based on the temporal context this is called *temporal aggregation*, i. e., data items are summarized into time intervals (granules such as days, months) according to their temporal affiliation. In *spatiotemporal aggregation* data tuples (data+time values) sharing the same spatial and temporal domain are grouped. For a survey on spatiotemporal aggregation the interested reader is referred to the work of López *et al.* [58].

According to Andrienko and Andrienko [4], data aggregation can be done either by calculating *data characteristics* (e. g., sum of data values, arithmetic mean and variance, minimum or maximum values) or by *grouping techniques* such as clustering or binning. Note that when simplifying the data, important information (e. g., outliers or anomalies that differ from the general data trend) can get lost, which is not desirable. In the following two grouping techniques, namely *clustering* and *binning*, will be briefly discussed:




(a)

(b)

**Figure 2.6:** Cluster-based visualization: (a) Calendar View of the number of employees (image taken from van Wijk and van Selow [93]); (b) ThemeRiver for visualizing time-dependent climate data (from Nocke et al [65]).

**Clustering:** Dividing data into a certain number of aggregations (clusters) of similar distance is called *clustering*. The goal is that data items within the same cluster are as similar as possible, while data items from different clusters are as different as possible. Depending on the particular data, different *similarity and distance measurements* can be applied (see Xu and Wunsch [95] for a survey of clustering algorithms). Known visualization approaches for clustered time dependent-data are for instance the *Rectangular View* [66], where clustered data is arranged as color-coded squares according to their temporal position allowing rearrangement to investigate time patterns, or the Clustered Calender View and the ThemeRiver which will be briefly presented and discussed in the following:

The *Cluster Calendar View* [93] is an example for the visualization of clustered time-dependent data. The data evolution over a certain time period (e. g., day or month) is grouped into several clusters which each represent the respective pattern. In figure 2.6 (a) the number of employees over a day is aggregated into the most significant seven clusters, each visualized by a line graph representing the clusters mean value (right). The clusters are also color coded in the calendar representation (left). This allows the observer to see the frequency of occurrence of each cluster while also seeing daily trends and patterns.

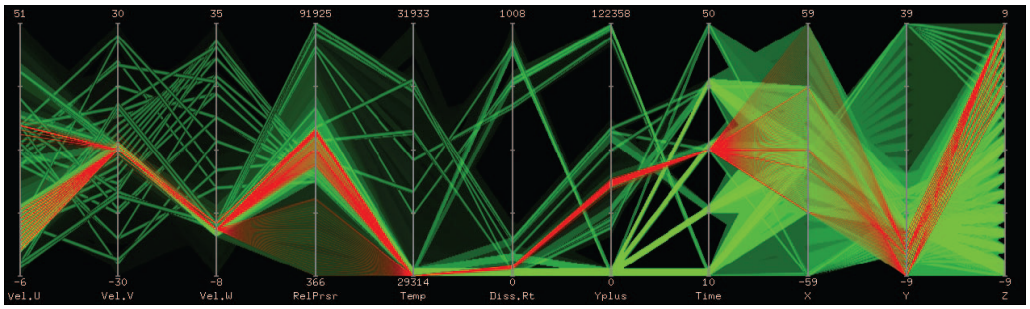


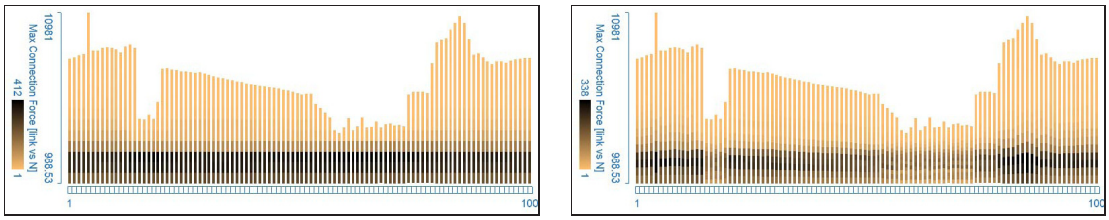

**Figure 2.7:** Outlier-preserving focus+context visualization of flow simulation data: brushed data items are highlighted in color (red polylines), the context information is depicted in green. Major trends and several outliers are well visible. Image taken from the work of Novotny and Hauser [67].

Additional interaction techniques such as brushing (and highlighting) of certain clusters or the visualization of the values of a single day are also possible with this approach.

In the work of Nocke et al. [65] a simplified version of the *Theme River* [33] approach (which was previously described in Sec. 2.2.1) was used to visualize the centroids of clusters for related parameters of climate data, which were normalized to the interval $[0, 1]$. Figure 2.6 (b) shows the visualization of the normalized cluster centroids for five parameters of meteorological data from the Potsdam observation station over 100 years. The parameters are based on several characteristics of the respective summer: p1: "total heat—sum of daily max. temperatures $t_{max} \geq 20C$" (orange); p2, p3: number of hot days where $t_{max} \geq 25C$ and $\geq 30C$, respectively (green, purple); p4: "summer mean temperature of daily mean temperatures" (dark green); and p5: mean of extreme temperature (yellow) [65]. A thin river band indicates a cold summer, in contrast, a broad river characterizes an extremely hot summer. Also major trends are well represented in the ThemeRiver. As the parameters are not equally treated (e. g., those near the middle are less distorted), interaction techniques such as parameter re-arrangement are integrated.

**Binning:** Binning divides the data space into several intervals (*bins*), each data item that belongs to a bin increases the respective occupancy value (e. g., histogram). Therefore binning gives a frequency-based representation of the data. *BinX* [7] uses binning along the time axis (temporal aggregation) at different levels of aggregation (dynamic aggregation) to display long time series (e. g., 50.000 timesteps). The mean, minimum and maximum value, and the standard deviation per bin are visualized to give the observer visual cues about the distribution of the aggregated data.

Novotny and Hauser [67] use two-dimensional *bin maps* for 2D data aggregation in parallel coordinates, in conjunction with outlier detection and clustering in the bin maps. This allows the interactive *focus+context visualization* (the concept was described in section 2.1.2) of large data sets, showing data trends while preserving outliers (see Fig. 2.7). For this purpose, an *output-oriented approach* is used, where only those parts of the visu-




(a) (b)

**Figure 2.8:** Segmented CurveViews [48]: (a) Global binning creates equal size bins for all time step, which facilitates direct comparison. (b) Local binning creates the same number of bins for each individual time step (according to the respective min./max. values), which reveals more details in dense populated areas.

alization process are executed which affect the final image. Although Novotny and Hauser apply this technique to parallel coordinates, it is also suitable for spatial data aggregation over time. Our visualization approach is inspired by the bin map technique, which will be explained in detail in section 3.2.

Konyha et al. [48] use *Segmented CurveViews* for the visualization of time-oriented data where vertical binning is applied. Thereby, each time step is handled individually in order to avoid misleading suggestions of continuity of the data in-between. The number of time series aggregated per bin is represented using color coding (see Fig. 2.8 where black corresponds to the largest and orange to the smallest number of aggregated curves). The user can decide whether (a) global or (b) local binning is applied. Accordingly, the respective vertical interval, which is subdivided into a certain number of equally sized bins, is defined by the minimum and maximum data value (a) over all time steps or (b) for each individual time step, respectively. Optionally, empty bins can also be hidden. For analysis purpose, brushing techniques and multiple linked views are combined, e. g., using LineBrushes in CurveViews [49], which is described later in Section 2.2.3.

### Further related approaches

Further approaches related to the visualization and analysis of time-dependent information are, e. g., symbolic representations or pixel-based techniques: *VizTree* [56] transforms the time series into a *symbolic representation*, encoding data in an augmented suffix tree. Here, global and local structures of time series data are summarized in a compact overview image. The user can interactively explore the data, by adding more details and discover non-trivial patterns. Moreover, space filling *pixel-based techniques* (compare to Keim [39]) aim to represent a maximal number of data items on the available amount of screen space. The work of Hao et al. [28] is such an example were time series data is represented at multiple resolutions according to degree-of-interest (DOI) values (also used in SimVis). Other related approaches are discussed in the following in the context of data selection (i. e., feature specification).

### 2.2.3 Time-Dependent Feature and Event Specification

As it was already mentioned, *features* are subsets of data which follow certain user-defined constraints. According to Post et al. [70], feature extraction can be done (semi-)automatically (e. g., [69]) or manually, for instance via *brushing* (e. g., Doleisch et al. [16, 18, etc.]). The later was briefly described in conjunction with SimVis in section 2.1.2.

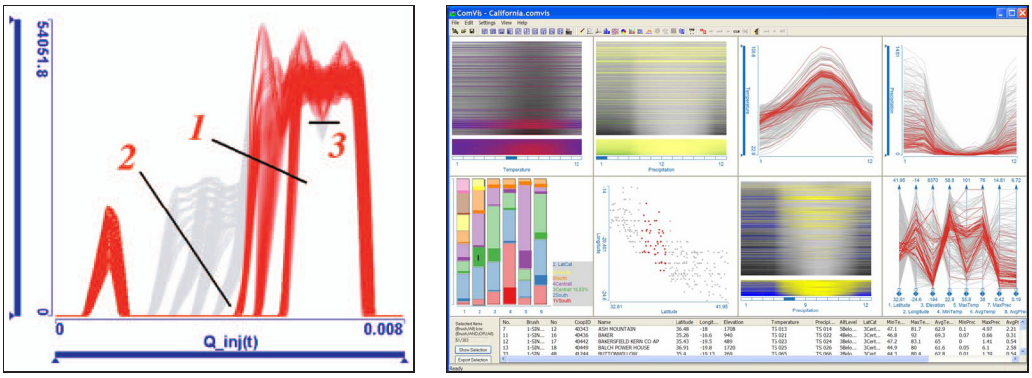
In the context of time-dependent data, the term *event* is commonly used for happenings of special interest in the evolution of data over time which trigger some automatic reaction (compare to the work done by Tominski [84]). Examples for *event types* are: the body temperature rises over $37°C$ (this can be evaluated at any time step); or the oil-prices increases more than 2 cent between three successive days (here a pattern is specified over a time sequence). After the event is specified, the time series is queried to find matches (i. e., detect the *event instances*) and the result is then represented in the visualization, called *event representation* (compare to Aigner et al. [1]).

Feature and event based visualization has proven to be a useful technique when representing huge data sets, e. g., by excluding (filtering) uninteresting data from the visualization (e. g., Hochheiser and Shneiderman [36]) or via focus+context visualization [30]. In the following, selected brushing and querying techniques for time series data are briefly presented and discussed. Note that these techniques can be considered as intuitive approaches for *event specification*. The interested reader is also referred to the work of Andrienko and Andrienko [4, Chap. 4.6], where several querying techniques for spatial and temporal data are presented and discussed.

#### Brushing and Querying Techniques for Time-Dependent Feature and Event Specification

As briefly described in section 2.1.2, feature visualization and specification via linking and brushing in multiple views is an integral part of the *SimVis* System and has also been used for the visual analysis of complex, time-dependent simulation data [19]. Raw data or derived higher-order attributes (e. g., velocity or local flow properties such as vorticity) are displayed and brushed in 2D views (e. g., scatterplots, histograms) and the respective features are visualized in other linked views (e. g., in a 3D view [62]).

*ComVis* [49, 48, 61, etc.] is another application providing multiple linked views (e. g., histograms, parallel coordinates, 2D scatterplots) and brushing functionalities for analysis purpose. In the context of time-dependent data, Konyha et al. [49] introduce **Line Brushes** to select function graphs (i. e., time series) intersecting with a simple line segment drawn in a *CurveView* (see Fig. 2.9 (a)). The time series are depicted as polylines, which is similar to the proposed approach. Using logical combinations of the line brushes enables the user to select outliers or to specify shapes of similarity (e. g., polylines have to intersect with several line brushes that are combined with a logical-AND, but may not intersect with other line brushes (logical-NOT)). Additionally, **Rectangular Brushes** can be used to select curves that enter and leave the rectangular brush at a certain edge




(a) (b)

**Figure 2.9:** Feature specification in *ComVis* [49, 48, 61, etc.] using multiple brushes in multiple linked views: (a) Logical combinations of several line brushes are used to select time series, which are represented as polylines in a CurveView. A selection is performed with brush 1, time series intersecting with the brushes 2 and 3 are subtracted (image taken from Konyha et al. [49]); (b) Composite brushing in multiple linked views including color lines views (left, top) curve views (right, top), a bar chart (left, bottom), a 2D scatterplot (middle, bottom) and parallel coordinates (right, bottom)—image taken from Matkovic et al. [61].

(e. g., to find a local maximum or minimum). In the context of time-dependent data, also *Segmented CurveViews* [48] (described in Sec. 2.2.2) and *Color Lines Views* [61] have be to be pointed out. The later represent each time series as a horizontal line (a row of pixels) in a rectangular view where the time-dependent data values are color coded (see Fig. 2.9 (left, top). Thereby, the time series (lines) are sorted according to their values at a user-defined time step.

The *TimeSearcher* [36, 35, 9, and others] is an application especially designed for the visual analysis of time series using several techniques from (information) visualization such as focus+context visualization, linking and brushing (in this context the term *dynamic querying* is used), and multiple views—these techniques were briefly explained in conjunction with *SimVis* in Sec. 2.1.2. By the use of (multiple) rectangular **Timeboxes** and/or **Angular Query Widgets** (compare to Hochheiser and Shneiderman [36, 35]) data items in the visualization can be brushed (i. e., selected data is visually highlighted) or filtered out (i. e., unselected data is removed). Using for instance angular query widgets, time series having a similar slope on a sequence of time steps are selected, i. e., the respective angles between the time series and the base line is within a certain range. The technique is inspired by **Angular Brushing** [32] which is used for brushing parallel coordinates in SimVis (see Fig. 2.10).

Figure 2.11 shows a screenshot from a further enhancement of the TimeSearcher [9], which allows the **Similarity-based Querying** for *temporal patterns* (events). This approach is similar to the brushing techniques presented in this thesis. In the bottom view an overview of the whole time sequence is given, in the middle and top view different

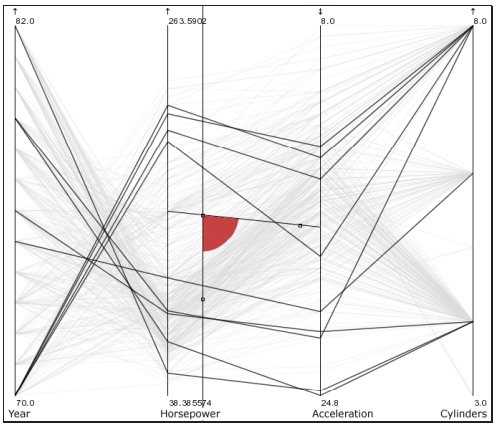



**Figure 2.10:** Angular Brushing is used to select line-segments which go down in-between the second and third axis (image taken from Hauser et al. [32]).

attributes are displayed. In this version of TimeSearcher, patterns can be specified by the user in three steps (compare to Buono et al. [9]): initially, the query scope is reduced by drawing timeboxes; then, a pattern is specified by selecting an item from the dataset (i. e., a polyline representing a single time series) and by restricting the pattern with a search-box (the red box in Fig. 2.11); a tolerance value can be specified and iteratively refined for the similarity-based queries. The start of each match is indicated by a red triangle, and the respective matching polyline is drawn in red (see Fig. 2.11). For similarity measurement Euclidean distance is used, additional transformations (e. g., offset translation, magnitude scaling, linear trend removal and noise reduction, see Keogh [41]) were implemented to make the matching process more robust. Recent developments allow for data driven forecasting by using river plots [10] which are similar to the ThemeRiver approach [33] previously presented.

**QuerySketch** [88] allows the user to sketch the shape of a pattern directly in the view, which is then used for similarity matching based on Euclidean distance. Inspired by this technique, **QueryLines** [77] allows the graphical specification of approximate queries of time series, where soft constraints and preferences are used for fuzzy pattern description and ranking of the query results, respectively. Curves that differ from the soft constraints increase their "penalty score" by the amount of violation. Four penalty types for QueryLines exist: minimum or maximum allow curves to pass above or below the line, respectively; the goal-type requires an exact match and any difference is penalized; the trend-type requires a similar direction (eg, increasing or decreasing). Additionally the lines can be chosen to be invariant against horizontal and/or vertical translation.

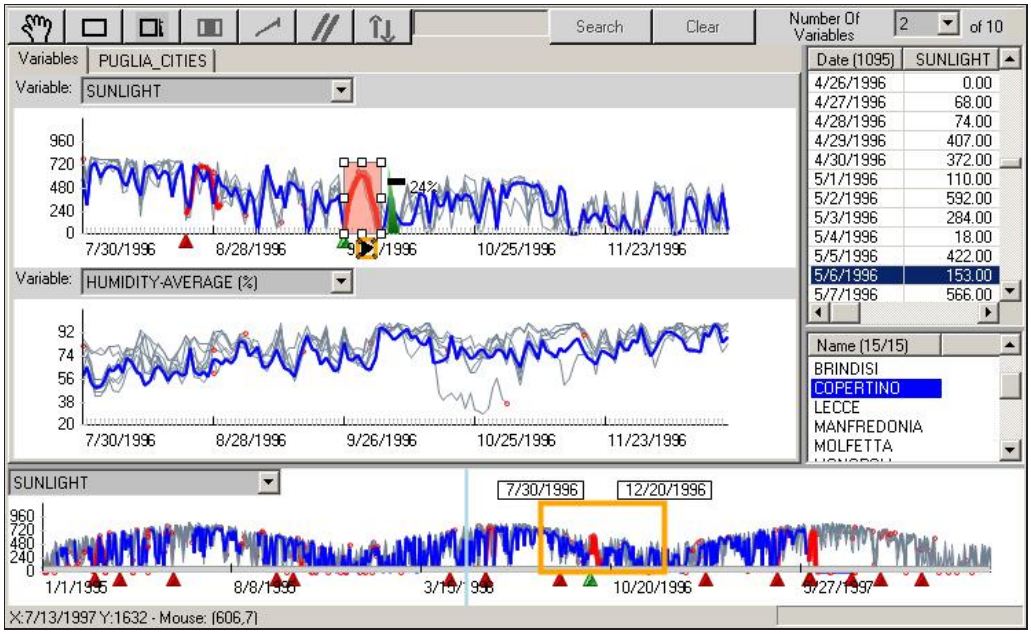



**Figure 2.11:** The TimeSearcher: (top, middle) Different attributes are mapped to the views. A pattern is specified (red box/curve in top view). All matches are indicated by a red triangle an a red polyline (top, middle); (bottom) An overview of the complete time series is given (image taken from Buono et al. [9]).

# 3 The CurveView

## Interactive Visual Analysis of large Time Series Data

In this chapter the visualization approach proposed in this thesis is described. Section 3.1 addresses the challenges and goals when depicting large amounts of time series data for visual analytics purpose. An outline on the visualization approach is given at the end of this section. The issue of data reduction is addressed in section 3.2 were also the applied technique is described. section 3.3 deals with the depiction of the binned data. Finally, the proposed visualization approach is reflected and discussed in section 3.4.

## 3.1 Challenges and Goals when Visualizing large Amounts of Time-Dependent Data

When analyzing time-oriented information, one is commonly interested in the evolution of the data over time. In the context of *interactive visual analysis*—also known as the science of *visual analytics* [82, 83] (see Sec. 2.1.1)—several issues have to be considered when depicting large amounts of time-dependent data. These issues were briefly described in the introduction (see Sec. 1.3), and in conjunction with the related state-of-the-art in section 2.2.2. They will be reconsidered in the following, motivating the visual analytics approach presented in this thesis:

- for the depiction of multi-variate data a *generic visualization approach* is desirable, which is suitable for different kinds and attributes of data;
- this approach has to enable the user to search for *visual structures* in the visualization, such as temporal *data trends* and visual *patterns*;
- therefore, the visualization approach has to deal with the issue of *clutter reduction* in the visualization; on the other hand, time series that differ highly from the trends, called *outliers*, still have to be represented in the visualization;
- and finally, *user interaction* (e. g., brushing and querying, zooming and panning) and the depiction of the data have to occur at interactive frame rates (i. e., *real-time visualization*).

### Using an intuitive and generic approach

According to Aigner et al. [2], a *generic approach* for the static representation of time-dependent data is to display polylines or curves, which interconnect the data values of a time series (compare to Sec. 2.2.1). This technique is well known from standard *line graphs* (compare to Harris [29]) and is commonly used due to its advantages (e. g., in the TimeSearcher [36] or the Cluster Calendar View [93]): The mapping of the discrete time steps, where the data values of a time series are given, to a *linear time axis* is very intuitive and self-explaining—thereby, time is considered to be a quantitative dimension. As a result, the technique is relatively easy to learn and comprehensible to a large audience. Moreover, the same generic approach can be applied to the visualization of different kinds and types of data, e. g., financial data, meteorological data, climate data, or simulation data, to name just a few.

On the other hand, when visualizing a multi-variate data set, this technique does not establish a direct visual connection between different attributes which evolve over time. This is due to the fact that commonly only one time-dependent attribute (e. g., one dimension) is depicted in a view. This issue is addressed in the SimVis approach by combining multiple *linked views* [8] visualizing different attributes of the data set. Thereby, the user is able to explore the interrelations between different attributes, e. g., by brushing data items in one view and observing the associated data highlighted in the other views (and dimensions). According to Aigner et al. [2]), the application of multiple linked views is beneficial for the visual analysis of time-dependent data.

### Visual Structures and Visual Clutter Reduction in Static Images

When depicting the line segments which interconnect the time-dependent data values in a line graph (described above), the primitives cross and overlap each other. Thereby, they form *visual structures* in the visualization, e. g., *visual data trends*, or *patterns* (see Fig. 3.1 (b)). These structures are significant for visual analytics, as they represent important information about the evolution of the data over time. Moreover, one is often interested in time series which do not behave like the general trends, called *outliers* or *anomalies*. Using these observations one can conclude *quantitative statements* about the visualized data (compare to Müller and Schumann [64] and Aigner et al.[2]).

In the visualization illustrated in figure 3.1 (b) and (c) the observer is able to compare different positions on the time axis, to search for the described visual data trends and to detect conspicuous visual patterns. However, the depiction of a large data sets with several hundred thousands of time series quickly ends in *overcrowded* and *visually cluttered displays*, as it is shown in figure 3.1 (a). Here, a data set with 30.930 time series given at 100 time steps is displayed using polylines, where each line is drawn with the same intensity. The amount of overlapping graphical primitives is much higher than the visually discriminable gray levels available in the image. Therefore, it is very difficult for the observer to identify visual structures in dense regions of the depiction. On the other

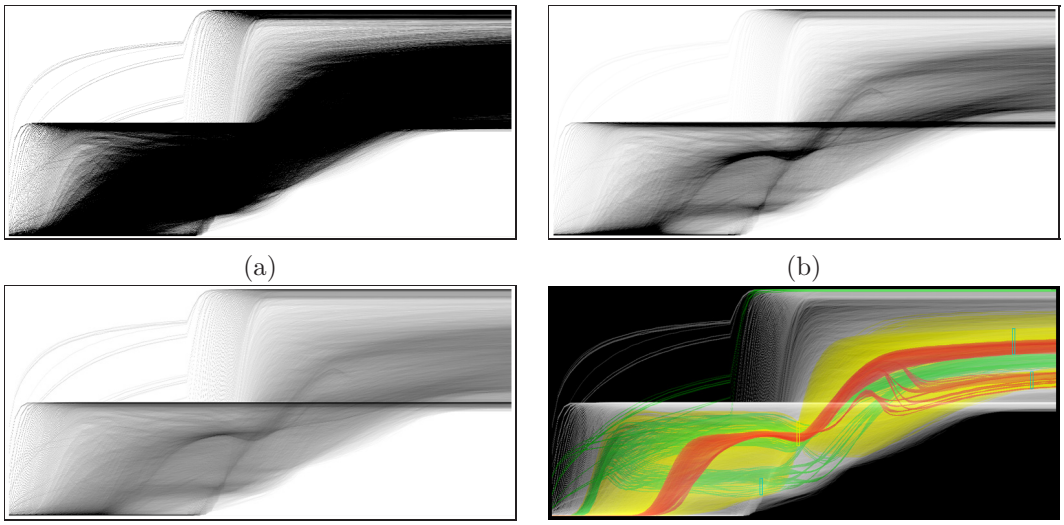


**Figure 3.1:** Visualization of 30.930 time series (curves) given at 100 time steps. (a) visual structures (i. e., general data trends) are hidden in dense regions by the massive amount of data; (b) focus is set on the trends, therefore, outliers in less populated areas disappear; (c) focus is set on the outliers while also trends are preserved; (d) features specified in multiple linked views are color coded.

hand, outliers are very well represented in this visualization, e. g., several single time series are visible in the top left area of figure 3.1 (a).

### Using Preattentive Graphical Features for Focus+Context Visualization

According to Fekete and Plaisant [23], *preattentive graphical features* (e. g., width, size, color (hue), intensity, etc.) are commonly used in information visualization for effective visualization of large data sets. This is due to the fact that these features "are processed by the [human] low level visual system and can be recognized 'at a glance' without effort" [23]. Preattentive graphical features are also well suited for *focus+context visualization* (see the work of Hauser [30]), where data items being interesting to the user (i. e., the *focus* portion of data) are visually emphasized, while the rest of the data (*context*) is displayed in a reduced style for orientation (see also Sec. 2.1.2). Such focus+context techniques are incorporated into the visualization approach presented in this thesis, as it is shown in figure 3.1 (b), (c), and (d).

In figure 3.1 (b) the focus is set on the general data trends, visible in the illustration. A linear mapping of the number of overlapping primitives (*density*) to the *intensity* values in the visualization is applied. In other words, the semi-transparent primitives are depicted using a *transfer function*[1], which maps the density values to the opacity (or transparency)

[1] Transfer functions are commonly used in volume visualization (VolVis) for nonlinear mapping of density values to opacity values in the 3D volume (e. g., Computed Tomography (CT) Scan) [55].

values of the primitives (compare to the work done by Johansson et al. [37]). However, the individual time series which diverge from the trends (i. e., *visual outliers*) and are visible in the top-left area in figure 3.1 (a) have disappeared in (b) because of the linear scaling. Here, the intensity of these lines is to low to be visible in the image.

In contrast to the linear mapping, a logarithmic mapping is applied in figure 3.1 (c), which emphasizes outliers while showing general data trends. Thereby, the linear interrelation of the intensity and density values does no longer exist, i. e., a pixel with twice the intensity of another pixel does not represent twice the amount of primitives passing through the pixel. According to Fekete and Plaisant [23], however, *transparency* is "not sufficient by itself to understand the number of overlaps. [. . . ] Therefore, transparency is only useful when it can be varied interactively to reveal overlaps and density of overlapping items".

Figure 3.1 (d) shows features specified in multiple linked SimVis views (and the CurveView itself) using *importance-driven color coding*—this coloring is applied in all SimVis attribute views and was briefly described in section 2.1.2. Thereby, user-defined colors are assigned to the different hierarchies in the feature definition language (FDL) tree (i. e., feature description, component and set). For instance, the green curves are brushed in another CurveView (feature set), the red time series are selected in the CurveView itself and in another 2D scatterplot view (feature component), whereas the time series depicted yellow are only selected in the CurveView (feature desciption). The process of this focus+context visualization of the time series will be detailed later in section 3.3.

### Real-Time Visualization and Interaction

When really large amounts of data (e. g., data sets containing several hundred thousand or millions of data entries) need to be depicted the process can take up to a few seconds, even on modern graphics hardware. In this context, *user interactions* such as navigating through the displayed information (e. g., zooming or panning), manual data selection (e. g., brushing [75], dynamic querying [36], or event specification [84]), can take relatively long. This is, however, not sufficient for the purpose of visual analytics of time-dependent data, where, in general, interactive frame rates are required (compare to the work done by Aigner et al. [1, 2]).

According to Keim's *visual analytics mantra* [40], techniques for (semi-)automated *data reduction* and *abstraction* (e. g., temporal data aggregation [58], or time-series classification [71]) have to be applied before visualization. These techniques transform the data into a compressed but still meaningful representation, which can then be visualized more efficiently in real-time. The data reduction process applied in the presented visualization approach uses 2D *bin maps* (compare to Novotny and Hauser [67]), which will be described in section 3.2.

#### Summary and Approach Outline

Different dimensions of multi-variate time-dependent data can be visualized using a relatively intuitive and easy to learn generic approach. Focus+context visualization enables the user, for instance, to visually assess how many time series run through a certain region on the screen (see Fig. 3.1 (b), (c) and (d)), or how relevant these time series are (see Fig. 3.1 (d) where importance-driven color coding is applied). Visual structures such as patterns and/or trends are visible even in overcrowded areas of the image—depending on the parametrization done by the user. Moreover, visual outliers (e. g., single time series or low populated areas) are still represented in the visualization. In addition to the visual issues, data reduction using binning is applied, transforming the time series data into a compressed but still meaningful representation which can be visualized in real-time.

## 3.2 Real-Time Visualization of large Time Series Data via 2D Binning

The visualization approach presented in this thesis uses 2D *bin maps* (compare to Novotny and Hauser [67]) for the aggregation of the lines connecting the time-dependent data values given at two successive time steps. This allows the depiction of the reduced data representation at interactive frame rates (i. e., in real-time). Additionally, the *degree-of-interest* (DOI) values [27] associated with the data items are aggregated into *DOI bin maps*, as these values are required for *focus+context visualization* in SimVis (compare Doleisch et al. [15, 17, 16, and others] and Hauser [30]). Instead of visualizing the raw data, the binned data is then depicted in real-time which will be described later in section 3.3. In doing so, the visualization is independent of the number of time series in the data set, which can be very large (e. g., several hundred thousand or even millions of time series), but dependent on the resolution of the (DOI) bin maps, which is determined by the user. In the following data aggregation using 2D bin maps (see Sec. 3.2.1) and DOI bin maps (see Sec. 3.2.2) is detailed.

### 3.2.1 Time Series Data Aggregation using 2D Bin Maps

In *data aggregation*, raw data is merged into smaller subsets by an aggregate function reducing the data volume [58] which can be used for means of interactive visualization (see the examples given in Sec. 2.2.2). In this context, *binning* is an aggregation technique where data items are transformed into a frequency distribution (e. g., histogram) by subdividing the data domain covered by the data values into a sequence of non-overlapping intervals called *bins*. A value is assigned to each bin (*bin count*), counting the number of data items that belong to it. In the following the procedure of building two-dimensional *bin maps* [67] from the time series data will be described and formalized (compare to the work done by Novotny and Hauser [67]).

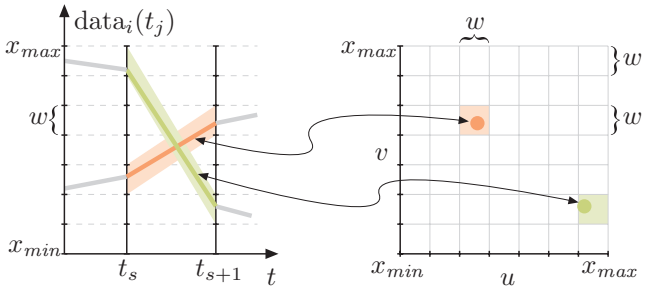



**Figure 3.2:** The interval $[x_{min}, x_{max}]$ is subdivided into $L = 7$ bins of equal width $w$. The line segments connecting the data values $x_{i,j} = \text{data}_i(t_j)$ given at two successive time steps $t_s$ and $t_{s+1}$ in (a) are transformed into a 2D *bin map* $\mathcal{M}_s$ in (b). The $x_{i,s}$ values given at time step $t_s$ are mapped to the $u$-axis, and the $x_{i,s+1}$ values at $t_{s+1}$ are mapped to the $v$-axis in the bin map, respectively. The illustration is adapted from Novotny and Hauser [67].

### Constructing a 2D Bin Map between two successive Time Steps

The visualization approach presented in this thesis applies two-dimensional *bin maps* [67] with each $L \times L$ bins—the resolution of the map, determined by the user—for aggregating the line segments which connect the data values of the time series between two consecutive time steps in the data set, namely $t_s$ and $t_{s+1}$. These maps can be thought as 2D histograms of the distribution of these line segments. When binning the data the interval $\mathcal{I} = [x_{min}, x_{max}]$ (see Eq. 1.3 in Sec. 1.2.1), containing all time-dependent data values of a data set $\mathcal{S}$, is subdivided into a set of $L$ non-overlapping bins $\mathcal{B}_k$ $(k = 1, \ldots, L)$ of equal bin width $w = \frac{x_{max} - x_{min}}{L}$, which is illustrated in figure 3.2, and can be expressed as

$$\mathcal{I} = [x_{min}, x_{max}] = \underbrace{[x_{min}, x_{min} + w[}_{\mathcal{B}_1} \cup \underbrace{[x_{min} + w, x_{min} + 2w[}_{\mathcal{B}_2} \cup \ldots \cup \underbrace{[x_{max} - w, x_{max}[}_{\mathcal{B}_L}.$$

In this context, a bin map $\mathcal{M}_s$ is a two-dimensional function of integer coordinates $(u, v)$ which counts the number of time series $\mathcal{D}_i \in \mathcal{S}$ (see Eq. 1.2) where the respective data values $x_{i,s} = \text{data}_i(t_s)$ given at the time step $t_s$ (see Eq. 1.1) are contained in the interval $\mathcal{B}_u$ (bin), and the data values $x_{i,s+1} = \text{data}_i(t_{s+1})$ given at an subsequent time step $t_{s+1}$ belong to the bin $\mathcal{B}_v$. The resulting set of time series fulfilling the described property between the time steps $t_s$ and $t_{s+1}$ is denoted as $\mathcal{S}_{match_s}(u, v)$. The *bin count* of each discrete entry (bin) in the map $\mathcal{M}_s$ can be formally described as

$$\mathcal{M}_s(u, v) = \text{bin\_count}_s(u, v) = \Big| \underbrace{\{\mathcal{D}_i \in \mathcal{S} \mid \text{data}_i(t_s) \in \mathcal{B}_u \wedge \text{data}_i(t_{s+1}) \in \mathcal{B}_v\}}_{\mathcal{S}_{match_s}(u,v)} \Big|, \qquad (3.1)$$

with $u = 1, \ldots, L$ and $v = 1, \ldots, L$, where $L \times L$ is the resolution of the bin map determined by the user, and $\mathcal{B}_u$ and $\mathcal{B}_v$ are the associated bins (compare to Novotny and Hauser [67]).

The principle of constructing a bin map is illustrated in figure 3.2: In (a) two time series are represented as polylines, each connecting the corresponding data values of a time series given at discrete time steps over the time axis $t$. In (b) the bin map is represented,

aggregating the line segments (e. g., green, orange) between the adjacent time steps $t_s$ and $t_{s+1}$ in (a). Each line segment in (a) can be mapped to a certain point with continuous coordinates $(u, v)$ in (b) and vice versa—in projective geometry this property is known as *duality*. Accordingly, the green line segment can be mapped to the green point and the orange line segment to the orange point.

The interval $[x_{min}, x_{max}]$ is subdivided into $L = 7$ bins of equal width $w$, thus the bin map in figure 3.2 (b) consists of $7 \times 7$ ($L \times L$) bins. Each bin represents the number of lines that belong to the respective parallelogram between the adjacent time steps in figure (a), e. g., the light green rhomboid contains the green line segment and the light orange one comprises the orange line segment. Therefore, the bin map is a discrete representation of the frequency distribution of the coordinates $(u, v)$ of these points in the map (b) and is just as well a discrete frequency-based representation of the line segments within the parallelograms in (a).

### 3.2.2 Aggregating Users Interest into 2D DOI Bin Maps

As it was already described in section 2.1.2, *SimVis* uses several well established concepts from visual analytics: One of them is the usage of fractional *degree-of-interest* (DOI) values (compare to Furnas [27]) from the unit interval which are associated with each data item in a SimVis view and represent a measurement of user's interest in a subset of the data. DOI values also build the basis for *focus+context visualization* [30] which is integrated in all (linked) SimVis views to visually discriminate interesting from uninteresting portions of the data (*feature visualization*). Thus the DOI information has to be preserved during the binning process as it is required for visualization. This will be described in the following:

In a SimVis view being part of the *feature definition language* (FDL) tree [16], three types of DOI values are associated with each data item $x_{i,j}$ of a time series (see Sec. 2.1.2): the values $doi_{i,j}^{comp}$ are associated to the respective *feature component* node in the FDL tree; the values $doi_{i,j}^{desc}$ belong to the particular *feature description* node; and the values $doi_{i,j}^{set}$ are related to the respective *feature set* node in the tree. These values build a DOI vector $\mathbf{doi}_{i,j}$ determined by a function

$$\mathrm{doi}_i(t_j) = \mathbf{doi}_{i,j} = (doi_{i,j}^{comp}, doi_{i,j}^{desc}, doi_{i,j}^{set})^{\mathrm{T}}, \quad \text{where } \mathbf{doi}_{i,j} \in [0,1]^3, \tag{3.2}$$

which assigns $\mathbf{doi}_{i,j}$ to each time-dependent data element $x_{i,j} = \mathrm{data}_i(t_j)$ in a view. Consequently, each tuple in a time series $\mathcal{D}_i$ from the data set in equation 1.2 is enhanced with a DOI vector $\mathbf{doi}_{i,j}$, i. e.,

$$\mathcal{D}_i' = \{(t_1, x_{i,1}, \mathbf{doi}_{i,1}),\ (t_2, x_{i,2}, \mathbf{doi}_{i,2}), \ldots, (t_N, x_{i,N}, \mathbf{doi}_{i,N})\}.$$

Similar to the approach of aggregating the number of line segments between two successive time steps $t_s$ and $t_{s+1}$ in a bin in the map $\mathcal{M}_s$ (see Eq. 3.1), the DOI values belonging to the respective line segments in $\mathcal{S}_{match_s}(u, v)$ in (3.1) are averaged in a bin. As the DOI

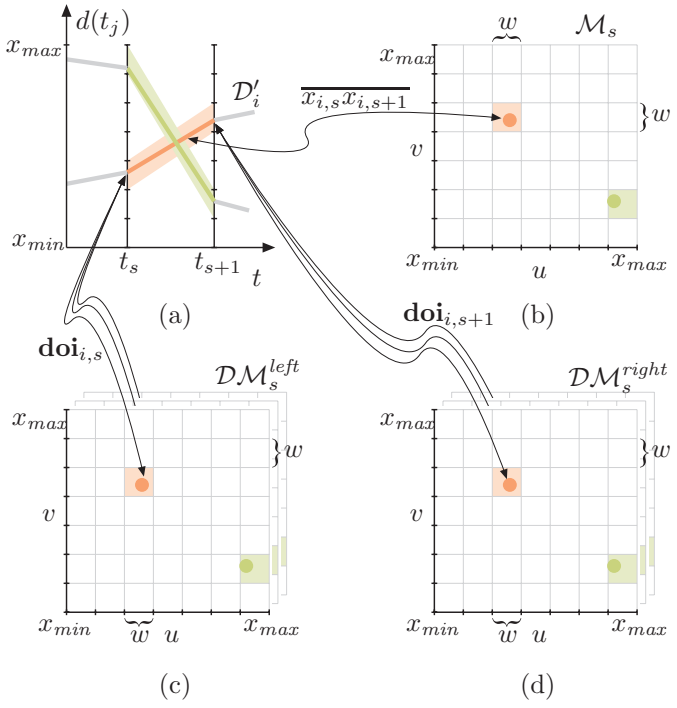



**Figure 3.3:** Aggregating the number of line segments and the associated DOI values in a bin map and two DOI bin maps, respectively: The interval $[x_{min}, x_{max}]$ is subdivided into $L = 7$ bins of equal width $w$. The line segments connecting the data values $x_{i,s}$ and $x_{i,s+1}$ given at two successive time steps $t_s$ and $t_{s+1}$ of a time series $\mathcal{D}'_i$ (a) are transformed into a 2D *bin map* $\mathcal{M}_s$ (b). The associated DOI values $\mathbf{doi}_{i,s}$ given at time step $t_s$ are averaged in a bin of the *DOI bin map* $\mathcal{DM}_s^{left}$ (c), and the DOI values $\mathbf{doi}_{i,s+1}$ given at the subsequent time step $t_{s+1}$ are averaged in a bin in the $\mathcal{DM}_s^{right}$ (d). Each of the DOI bin maps (c, d) consists of three layers for the different types of $\mathbf{doi}_{i,j} = (doi_{i,j}^{comp}, doi_{i,j}^{desc}, doi_{i,j}^{set})^{\mathrm{T}}$.

values $\mathbf{doi}_{i,s} = \mathrm{doi}_i(t_s)$ and $\mathbf{doi}_{i,s+1} = \mathrm{doi}_i(t_{s+1})$ given at the time step $t_s$ and $t_{s+1}$, respectively, need not be equal, they have to be subsumed in separate *DOI bin maps*, namely $\mathcal{DM}_s^{left}$ and $\mathcal{DM}_s^{right}$. These DOI maps can be considered as two-dimensional functions

$$\mathcal{DM}_s^{left}(u,v) = \begin{pmatrix} doi_{u,v}^{lcomp} \\ doi_{u,v}^{ldesc} \\ doi_{u,v}^{lset} \end{pmatrix} = \frac{1}{\mathcal{M}_s(u,v)} \sum_{\mathcal{D}_i \in \mathcal{S}_{match_s}(u,v)} \mathrm{doi}_i(t_s) \quad \text{and} \tag{3.3}$$

$$\mathcal{DM}_s^{right}(u,v) = \begin{pmatrix} doi_{u,v}^{rcomp} \\ doi_{u,v}^{rdesc} \\ doi_{u,v}^{rset} \end{pmatrix} = \frac{1}{\mathcal{M}_s(u,v)} \sum_{\mathcal{D}_i \in \mathcal{S}_{match_s}(u,v)} \mathrm{doi}_i(t_{s+1}), \tag{3.4}$$

where $\mathcal{M}_s(u,v)$ is the bin count and $\mathcal{S}_{match_s}(u,v)$ is the set of matching time series, both defined in (3.1), between the time steps $t_s$ and $t_{s+1}$. The coordinates of the elements in the DOI bin maps $\mathcal{DM}_s^{left}$ and $\mathcal{DM}_s^{right}$, as well as the coordinates in the bin map $\mathcal{M}_s$, are denoted as $u = 1, \ldots, L$ and $v = 1, \ldots, L$, where $L \times L$ is the resolution of each map. Hence, a number of $3L^2$ bins are required for one DOI bin map to aggregate the tree

DOI values of the vectors. In figure 3.3 the process of constructing a bin map and two DOI bin maps is illustrated.

Aggregating the whole data set $\mathcal{S}$ including DOI information (i. e., all time series $\mathcal{D}'_i \in \mathcal{S}$ over all time steps $t_j$ where $j = 1, \ldots, N$) means constructing a bin map $\mathcal{M}_s$ and two DOI bin maps $\mathcal{DM}_s^{left}$ and $\mathcal{DM}_s^{right}$ for each pair of consecutive time steps $t_s$ and $t_{s+1}$ where $s = 1, \ldots, N-1$, and $N$ is the number of time steps per time series[2]. Accordingly, the binning approach uses a number of $(N-1) \cdot (L^2 + 2 \cdot 3L^2) = (N-1) \cdot 7L^2$ bins for the aggregation of the data set[3]. The visualization of the binned data instead of the raw data (see Sec. 3.3) is therefore independent of the number of time series $\mathcal{D}'_i \in \mathcal{S}$, which can be very high (e. g., several hundred thousand or even millions of data entries), but is dependent on the resolution of the bin map and DOI bin maps determined by the user.

## 3.3 Focus+Context Visualization Based on Binned Data

In section 3.2 it was just described how 2D *bin maps* [67] and *DOI bin maps*, which aggregate the time dependent data values and the associated *degree-of-interest* (DOI) values (compare to Furnas [27]), are built, both given at two successive time steps in the time series data. This section deals with the issue of visualizing the binned data instead of the original data, including the degree-of-interest information, and using focus+context visualization. As it was briefly described in section 3.1, *preattentive graphical features*, such as size, width, color (hue) or transparency (intensity), are commonly used for effective visualization of large data sets [23]. These features are well suited for *focus+context visualization* (compare to Hauser [30]), where interesting data items (focus) are visually highlighted while the rest of the data (context) is depicted only in a reduced style for orientation.

The idea of the proposed visualization approach is to reverse the binning process, i. e., to draw for each non-empty bin in the bin map $\mathcal{M}_s$ the associated parallelogram containing all possible line segments which could have been aggregated in the 2D bin during the binning process (compare to the work of Novotny and Hauser [67] and see Fig. 3.2). Moreover, the bin count is mapped to some preattentive visual feature of the primitives (e. g., intensity/transparency). This allows the observer to easily count or estimate the number of time series—represented as polygons—that pass through a certain region in the visualization. Therefore, it is important that the user can interactively change the transparency mapping to reveal the number of overlapping items (compare to Fekete and Plaisant [23]). In addition to that, the DOI values from the DOI bin maps $\mathcal{DM}_s^{left}$ and $\mathcal{DM}_s^{right}$ belonging to the line segments binned in $\mathcal{M}_s$ are represented by another visually discriminable feature. For this purpose, SimVis uses importance-driven color coding, where

[2] Note that each time series in the data set has the same number of time steps $N$.

[3] Giving an example: a time dependent data set, with $N = 100$ time steps, where the user selects the resolution of the maps $L \times L = 256 \times 256$, and 32 bit are used per bin count ($L^2$) and 8 bit per DOI bin ($2 \cdot 3L^2$), requires approximately 62 MB for aggregating the whole data set.

the DOI values are an importance measure for each data item (compare to the work done by Doleisch et al. [15, 17, 21, etc.]).

#### Representing Visual Structures for Visual Analytics

To reduce visual cluttering, Johansson et al. [37] use high-precision *density maps* (compare to Wegman and Lou [89]) in their work on parallel coordinates to count the number of primitives that pass though each element on the screen (pixel). During the visualization an *opacity value* is mapped to each element in the density map, utilizing a user-defined *transfer function*[4] to overcome the restricted precision of the output device (e. g., only a limited number of different intensity levels are available per pixel). Using this nonlinear mapping, regions containing only few primitives can be visually emphasized, for instance, while general data trends can be suppressed, or trends can be highlighted while outliers are depicted with reduced opacity [37]. The visualization presented in this thesis uses a similar approach which additionally incorporates importance information (i. e., DOI values) about the displayed data.

### 3.3.1 Generating a DOI Density Map including Density and DOI Information

In the following the generation of high-precision *density maps* [89] is described, as it is also applied in the work of Johansson et al. [37] for the representation of structures in parallel coordinates visualization. In connection with feature representation using focus+context visualization in SimVis, the *degree-of-interest* (DOI) values [27] need to be incorporated into the density map—the resulting map will be denoted as *DOI density map* and combines both density and DOI information:

For our purpose, a high-precision DOI density map can be defined as a two-dimensional function of integer coordinates $\mathbb{N} \times \mathbb{N}$ on a set of values $\mathbb{P}$:

$$\mathcal{T}(u,v) = (\rho_{u,v}, \underbrace{d_{u,v}^{comp},\ d_{u,v}^{desc},\ d_{u,v}^{set}}_{\mathbf{d}_{u,v}})^{\mathrm{T}} \in \mathbb{P}^4 \quad \text{and} \quad u,v \in \mathbb{N}. \tag{3.5}$$

The idea is to represent each pixel on the display device with an element in the DOI density map $\mathcal{T}$ at the coordinates $(u, v)$, which each contains four entries (see also Fig. 3.4): (1) the number of geometric primitives that pass though that pixel divided by the overall number of time series—representing the *density value* $\rho_{u,v}$; and (2–4) the vector $\mathbf{d}_{u,v} = (d_{u,v}^{comp}, d_{u,v}^{desc}, d_{u,v}^{set})^{\mathrm{T}} \in \mathbb{P}^3$ which sums up the associated DOI values of the primitives running through the map entries. Note that the sum of DOI values is averaged later when depicting the map (see Sec. 3.3.2). Here the elements representing the feature component, the feature description and the feature set are contained in the vector (compare to Eq. 3.2). Thereby, the precision (i. e., codomain) of $\mathbb{P}$ has to be high enough, allowing the lossless

[4] Transfer functions are commonly used in Volume Visualization for nonlinear mapping of opacity values to density values in the 3D volume (e. g., Computed Tomography (CT) Scan) [55].

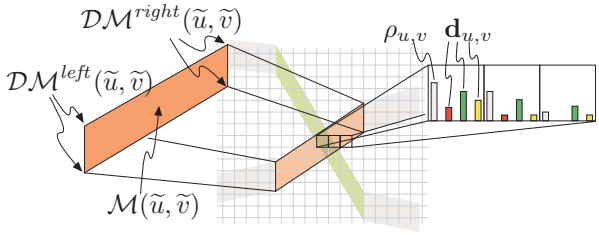



**Figure 3.4:** A *DOI density map* $\mathcal{T}(u,v)$ where an element is associated with each pixel on the screen, consisting of a *density value* $\rho_{u,v}$ (gray bar) and a vector $\mathbf{d}_{u,v} = (d_{u,v}^{comp}, d_{u,v}^{desc}, d_{u,v}^{set})^{\mathrm{T}}$ (red, green, yellow bars) which accumulates the three *DOI values* representing the different node types in the FDL tree in SimVis. The attributes of the orange parallelogram are determined by the bin count in the bin map $\mathcal{M}$ and the DOI values in the DOI bin maps $\mathcal{DM}^{left}$ and $\mathcal{DM}^{right}$, each given at the discrete coordinates of the bins $(\widetilde{u},\widetilde{v})$.

aggregation of all four component parts of the geometric primitives that pass through the respective elements in the DOI density map.

In figure 3.4 such a map is illustrated. An orange parallelogram is magnified, whose attributes are defined by the bins given at the coordinates $(\widetilde{u},\widetilde{v})$ in the bin map $\mathcal{M}$ and in the DOI bin maps $\mathcal{DM}^{left}$ and $\mathcal{DM}^{right}$, which will be described in the following:

- The vertical *coordinates* of the rhomboid are derived from $(\widetilde{u},\widetilde{v})$ and refer back to the vertical start and end position of the intervals (bins) $\mathcal{B}_{\widetilde{u}}$ and $\mathcal{B}_{\widetilde{v}}$ during the binning process (see Sec. 3.2). The horizontal coordinates represent the position of the successive time steps on a linear time axis. Note that the distance between these time steps is not equal if time is sampled in irregular intervals.
- The bin count given at $\mathcal{M}(\widetilde{u},\widetilde{v})$ is directly associated with the *weight* of the geometric primitive, as it represents the number of line segments that are aggregated in it. The weights of all parallelograms running through the elements in the DOI density map $\mathcal{T}(u,v)$ are summed up in the respective density values $\rho_{u,v}$. For implementation purpose, the respective division by the overall number of time series is assumed implicitly and performed as a final step.
- The averaged DOI values from the DOI bin maps $\mathcal{DM}^{left}(\widetilde{u},\widetilde{v})$ and $\mathcal{DM}^{right}(\widetilde{u},\widetilde{v})$ are each multiplied with the respective bin count $\mathcal{M}(\widetilde{u},\widetilde{v})$ to represent the accumulated DOI values of the time series aggregated in the respective bin. The multiplied values are then mapped to the corners (vertices) of the orange parallelogram (see Fig. 3.4). The DOI values at an arbitrary position within the geometric primitive are obtained by (bi-)linear interpolation of the values given at its vertices. The DOI values are summed up in the vectors $\mathbf{d}_{u,v}$ of the affected DOI density map entries $\mathcal{T}(u,v)$.

The usage of the high-precision DOI density map assures that no information in the data is lost, apart from the data which is lost during binning, of course. This means that even regions where only a few time series pass through or regions with small DOI values are both represented in the map.

### 3.3.2 Depicting the DOI Density Map using Focus+Context Visualization

After the DOI density map is constructed as it was just described, the following deals with the final rendering of the map which results in the depiction of the data on the output device: As the precision of the output device is usually lower than the map, only a limited number of colors (hues) and intensity levels can be displayed. Thus, one has to deploy techniques (e. g., focus+context visualization) to overcome this issue when visualizing huge data sets. Thus, the proposed approach maps the number of overlapping primitives (density) to opacity values, and DOI information to color values.

#### Mapping Density to Opacity Values

In the work of Johannson et al. [37] on parallel coordinates *opacity transfer functions* are used for non-linear mapping of the values in the *density map* to opacity/transparency in the final image—in volume visualization (VolVis) these functions are used from the very beginning for similar purposes (e. g., the work of Levoy [55] in 1988). Such an one-dimensional opacity transfer function can be defined as $\mathrm{tf}_\alpha : \rho \rightarrow \alpha$, i. e.,

$$\alpha(\rho) = \mathrm{tf}_\alpha(\rho), \quad \text{where } \alpha(\rho) \in [0,1] \text{ and } \rho \in [0, \rho_{max}] \subseteq \mathbb{P}. \tag{3.6}$$

Here, $\rho$ represents the density value (i. e., number of overlapping time series divided by the overall number) taken from the (DOI) density map $\mathcal{T}$, which has to be within the codomain $\mathbb{P}$ of the map. The resulting *opacity/transparency value* $\alpha$ is within to the unit interval, where 1 denotes completely opaque and 0 means fully transparent. This also corresponds with the minimal and maximal intensity value, which can be displayed by the output device. As the transfer function can be an arbitrary (non-linear) mapping, it allows to visually amplify certain parts of the density map (e. g., dense regions), while suppressing the visual appearance of other areas in the map. According to Hauser [30], the usage of an opacity transfer functions for data representation fits well with the requirements of focus+context visualization. Giving the user the facility to change the transfer function interactively, transparency is very useful to reveal the number of overlapping primitives (compare to Fekete and Plaisant [23]).

In the visualization approach presented in this thesis, two kinds of opacity mapping exist, namely *linear* and *logarithmic*, where the user can chose between the two. A linear mapping is illustrated in figure 3.5 (a), which focuses on the general data trends highlighted in figure (b). Thereby, the observer gets an intuitive (linear) impression of the amount of time series passing through certain areas. On the other hand, a logarithmic mapping (see Fig. 3.5 (c)) emphasizes the outliers, while preserving the data trends, as it is shown in figure (d). Both opacity transfer functions in figure (a) and (c) are dependent on an user-defined *scale factor* (see the different slopes of the orange line and curve, respectively). To ensure, that outliers are preserved in the visualization (independent of linear or logarithmic mapping), a constant *opacity offset* $\alpha_0 \in [0,1]$, specified by the user, is added to the

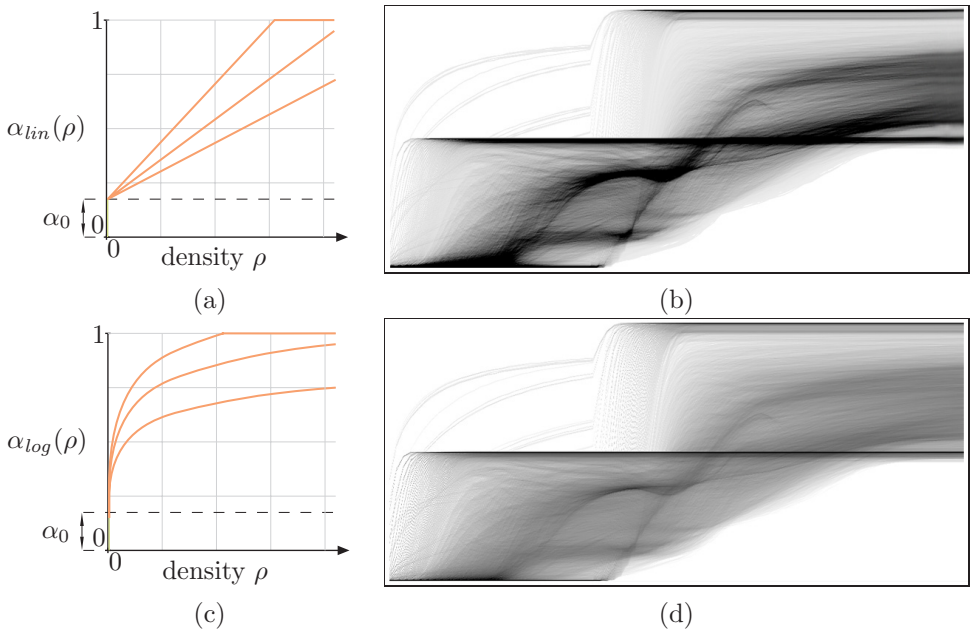



**Figure 3.5:** Mapping density $\rho$ to opacity values: (a) linear transfer function $\alpha_{lin}(\rho)$ with different slopes (i. e., scale factors) and constant offset $\alpha_0$; (b) trends are highlighted according to the scale factor, and outliers are preserved, because of the offset; (c) logarithmic transfer function $\alpha_{log}(\rho)$ using different scale factors (orange curves) and offset $\alpha_0$; (d) the outliers are highlighted, while preserving the trends.

respective opacity values (see Fig. 3.5 (a) and (c)). Note, that the functions are clamped at the maximal available opacity value (one).

When the resolution of the (DOI) bin maps is reduced, the average bin count increases proportionally, i. e., then more line segments are aggregated in a single bin, and vice versa. To keep the (linear or logarithmic) mapping of the transfer function $\mathrm{tf}_\alpha(\rho)$ (see Eq. 3.6) independent of the resolution of the (DOI) bin maps, the respective density value $\rho$ from the density map is multiplied with the number of bins currently used. Moreover, the user-defined scale factor is divided by the overall number of time series in the data set (i. e., for implementation purpose this division is performed as a final step), adapting the function to different amounts of data and in order to fulfill the requirements for the density values (i. e., number of overlapping time series divided by the overall number).

### Mapping DOI Values to Color Values for Feature Visualization

In conjunction with feature based techniques and *focus+context visualization* [30], SimVis uses *importance-driven color coding* using a transfer function $\mathrm{tf}_{doi}$ to represent the user's *degree-of-interest* (DOI) [27] in a subset of the displayed data briefly described in section 2.1.2. As this color coding is also incorporated in the visualization approach presented in this thesis, it will be detailed in the following:

In SimVis the user can assign an arbitrary color to each of the three kinds of feature

nodes in the hierarchical *feature definition language* (FDL) tree [16], i. e. to the feature component, the feature description, and the feature set node (see Sec. 2.1.2)—by default, the colors red, green, and yellow are assigned. Additionally, the user can decide whether the data is depicted on a black or a white background. Thereby, the base color, which is associated to data where the DOI values are close or equal to zero, is implicitly defined (i. e., white base color on black background, and black on white background).

For the *importance-driven mapping* of DOI values to Red-Green-Blue (RGB) *color values*, commonly used on display devices, a transfer function $\mathrm{tf}_{doi} : \mathbf{doi} \rightarrow \mathcal{C}_{RGB}$ is applied:

$$\mathcal{C}_{RGB}(\mathbf{doi}_{u,v}) = \mathrm{tf}_{doi}(\mathbf{doi}_{u,v}), \quad \text{where} \quad \mathcal{C}_{RGB},\ \mathbf{doi}_{u,v} \in [0,1]^3. \tag{3.7}$$

Here, $\mathcal{C}_{RGB}$ denotes the resulting RGB color values from the RGB *color space*[5] which is scaled to the unit cube. Furthermore, the three feature nodes in the FDL tree are represented in the values of the DOI vector $\mathbf{doi}_{u,v} = (doi_{u,v}^{comp}, doi_{u,v}^{desc}, doi_{u,v}^{set})^{\mathrm{T}}$, which is also contained within the unit cube (compare to Eq. 3.2). The transfer function $\mathrm{tf}_{doi}$ combines the colors assigned to the FDL tree nodes by the user, weighting each with the importance represented by the respective DOI value in $\mathbf{doi}_{u,v}$ as it is illustrated in figure 3.6 and detailed below. To avoid that the user-specified colors mix additively amongst each other (e. g., green + red becomes yellow when combined additively) the transfer function has to assure that more *important colors obliterate those less important*, according to the hierarchy of the associated feature node in the FDL tree. Thereby, the color assigned to the feature description node covers the color associated with the feature set node and both obliterate the color assigned to the feature component node[6].

The computation of the resulting color—considering the mentioned restrictions—is illustrated in figure 3.6: At the lowest hierarchical level in color coding, a linear interpolation is performed between the base and the feature component color. Thereby, the weight $\lambda_1$ is the difference of the respective DOI values of the feature component and the feature set node, which has to be greater or equal to zero (i. e., $doi_{u,v}^{set}$ reduces $doi_{u,v}^{comp}$ in order to avoid the mixing of the colors). Using the same principle, $\lambda_2$ is computed and used when interpolating between the previously computed color and the feature set color. Finally, the resulting color is calculated using another linear interpolation between the previously interpolated color(s) and the feature description color. As one can see from the figure, the DOI value of the feature description node has the highest influence on the final color. From top to bottom, each hierarchical level affects the lower hierarchies by subtracting the respective DOI values. Thus, the transfer function represents a non-linear color mapping. The concept will be illustrated in the following on the example of depicting the DOI density map.

[5] See the website of the *International Color Consortium* (ICC) http://www.color.org/

[6] Note that the hierarchy in color coding does not correspond to the hierarchy of the nodes in the FDL tree. This is due to the fact, that the feature description nodes are combined using a fuzzy OR-operation (in the feature set node), and the feature component nodes are combined using a fuzzy AND-operation (in the feature description node).

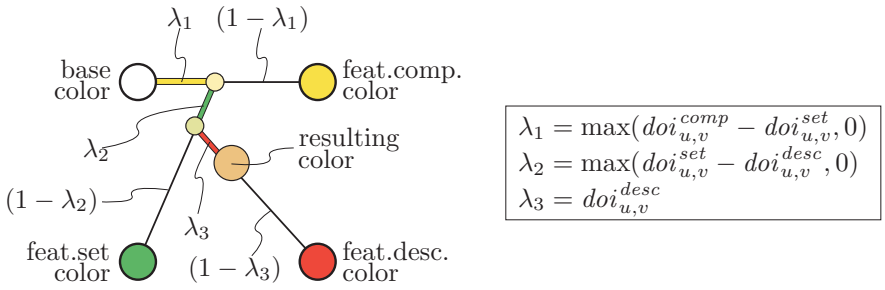



**Figure 3.6:** Importance-driven color mapping of the DOI values, which are associated to the nodes at different hierarchies in the FDL tree. Three (hierarchical) linear interpolations are performed between the respective colors, the corresponding weights are formalized in the right box.

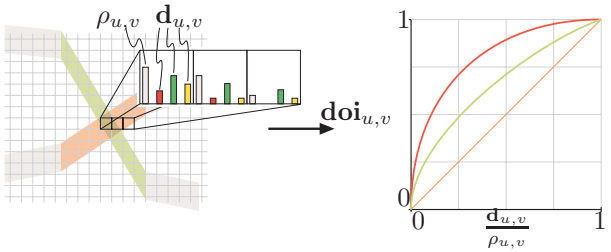



**Figure 3.7:** The values $\mathbf{d}_{u,v}$ and $\rho_{u,v}$, from the DOI density map $\mathcal{T}(u,v)$ (a) are used to calculate the respective enhanced DOI values $\mathbf{doi}_{u,v}$ in (b), using a (non-)linear gamma function (see Eq. 3.8). The cases were $\gamma = 1$ (straight orange line), $\gamma = 0.5$ (square root function, green), and $\gamma = 0.25$ (orange curve) are illustrated.

When visualizing the DOI density map $\mathcal{T}$, each element $\mathcal{T}(u,v)$ corresponds to a pixel on the output screen and contains three values in the vector $\mathbf{d}_{u,v}$ and a density value $\rho_{u,v}$ (see Eq. 3.5 and Fig. 3.7 (a)). When creating the map $\mathcal{T}$, the DOI values from the passing through geometric primitives are added up (see Fig. 3.4 in Sec. 3.3.1). Consequently, the values in the vector $\mathbf{d}_{u,v}$ are no longer restricted to the unit cube and therefore have to be normalized to the codomain of the DOI values $[0,1]$ before applying the transfer function $\mathrm{tf}_{doi}$, i. e., $\frac{\mathbf{d}_{u,v}}{\rho_{u,v}}$. Due to this normalization, however, it can happen that *DOI outliers* are not represented in the visualization. For instance, when a single time series with a maximal DOI value (= 1) is running through a region with several thousand time series with zero DOI (context). Then, the respective average DOI value would be $\frac{1}{1.000}$. Accordingly, the important time series would be hardly discriminable from the context. Thus, again a (non-)linear mapping is applied to calculate the (emphasized and) *normalized DOI vector* $\mathbf{doi}_{u,v}$ used for the importance-driven color mapping in equation 3.7:

$$\mathbf{doi}_{u,v} = \left(\frac{\mathbf{d}_{u,v}}{\rho_{u,v}}\right)^{\gamma}, \quad \text{where} \quad \mathbf{doi}_{u,v} \in [0,1]^3 \quad \text{and} \quad \gamma \in ]0,1]. \tag{3.8}$$

The respective function graphs for different gamma values are illustrated in figure 3.7 (b).

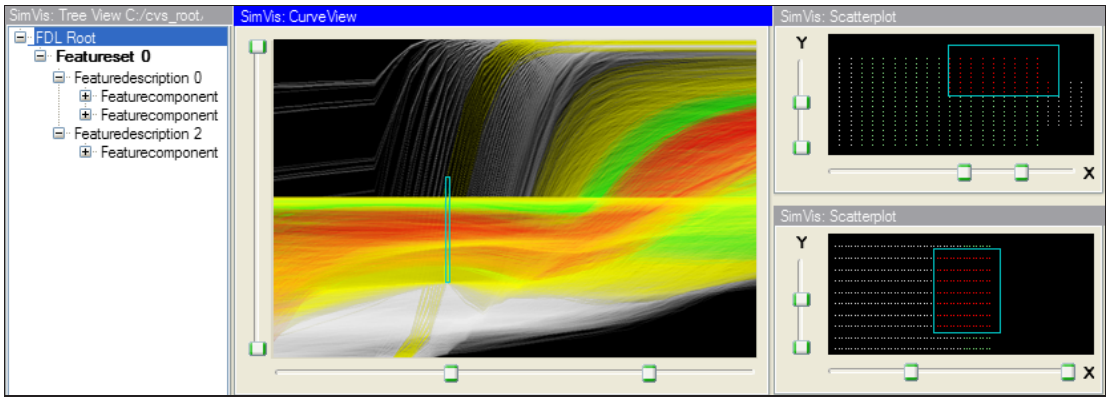



**Figure 3.8:** In SimVis different colors are assigned to the *feature set* (green), the *feature description* (red), and the *feature component* (yellow) nodes, which are organized in a hierarchical *feature definition language* (FDL) tree (left view). The *CurveView* (middle) and the *scatterplot view* (right, top) are both child-nodes of the same feature description node. The other scatterplot (right, bottom) belongs to another feature description node. All views represent feature component nodes, and are child-nodes of the same feature set node in the FDL tree. Complex features are specified by brushes (green rectangles) in the views, and are depicted using focus+context visualization via *importance-driven coloring* and DOI enhancement ($\gamma \approx 0.5$, middle).

Note that, however, no constant offset is is added to the DOI values in order to retain the (relative) ratios of the different feature nodes in the FDL tree (i. e., feature component, description and set). In the CurveView, the user can adjust the gamma value using a slider.

In figure 3.8 (middle), an example visualization of time series in a CurveView with features encoded in color is shown. Thereby, the color red is assigned to the feature description nodes, green to the feature set nodes and yellow to the feature component nodes in the FDL tree (left view). Note that each attribute view providing brushing functionality in SimVis (i. e., all views in Fig. 3.8) represents a feature component node (in the FDL tree). Data items, which are only selected in the CurveView (middle), are colored in yellow (representing the feature component node). Unselected data items (i. e., the respective DOI values are equal or close to zero) are depicted in a color close to the base color (white). The red highlighted time series are selected in the CurveView (middle) using a time step brush (represented as a green rectangle and described later in Sec. 4.2). The red curves are selected additionally—in other dimensions—in a 2D scatterplot view (right, top) using a smooth brush (green rectangle). Since the two views are both child-nodes (i. e., feature component nodes) of the same feature description node (see the FDL tree, left), the features specified in both views (the respective DOI values are combined in the FDL tree by a fuzzy AND-operation) are colored in red.

All displayed attribute views in figure 3.8 are also child-nodes of the same feature set node (green color assigned) where the DOI values from the feature description nodes are

combined using an implicit (fuzzy) OR-operation. Therefore, the data items, which are selected in the other scatterplot view (right, bottom) but are not brushed in the CurveView and the first scatterplot (right, top)—the latter are colored red—, are depicted in green color (middle view). To point out (and summarize) what was mentioned before, the color (e. g., red) assigned to the feature description node—here the feature component (child-)nodes are combined using a fuzzy AND-operation—obliterates the color (e. g., green) associated with the feature set node—here the feature description (child-)nodes are combined using a fuzzy OR-operation. Both cover the color assigned to the feature component node (e. g., yellow), i. e., the features specified only in the respective attribute view. Combining focus+context visualization and brushing in multiple views, organized in a hierarchical FDL tree, allows the user to specify and visually analyze complex features in the data.

## 3.4 Discussion of the Visualization Approach

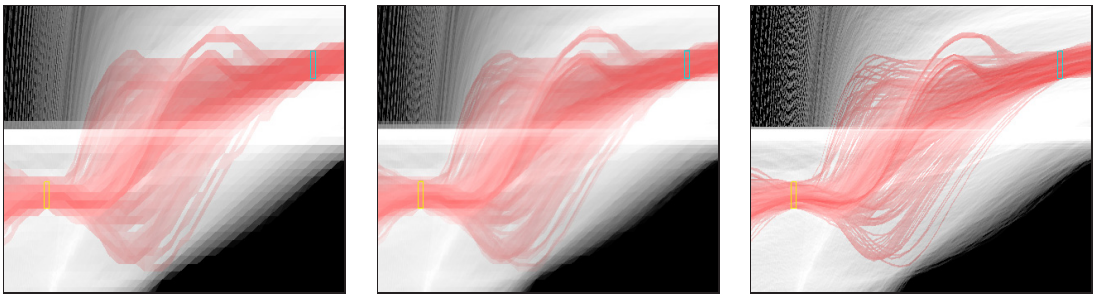
In the following, several aspects related to the visualization approach will be discussed briefly: these are issues regarding the binning approach such as the granularity (quality) of the visual output when zooming into the visualization (see Sec. 3.4.1); and different mappings, which accent certain aspects of the time-dependent data (e. g., trends, outliers or features) using focus+context visualization are compared in section 3.4.2.

### 3.4.1 Two-dimensional Binning

The 2D *bin map* approach (compare to Novotny and Hauser [67]) turns out to be very useful for the purpose of data reduction allowing the real-time visualization of a large number of time series (e. g., 500.000–1.000.000 time series given at about 70 time steps, see the performance evaluation in Sec. 6.3). Thereby, the rendering time for the binned data representation is rather independent of the number of time series in the data set, but dependent on the resolution of the bin maps, which is determined by the user, and the number of time steps in the visible area of the view.

#### Granularity of the Visualization

As this approach depicts parallelograms instead of polylines, connecting the original data values, the *granularity* of the visual output is highly dependent on the number of bins used per bin map and DOI bin map, respectively. Imagine the width $w$ of each 2D bin in the map would strive against zero, then the map contains a number of $N \times N$ bins, where $N \to \infty$. Then, roughly spoken, each drawn rhomboid associated to such a bin would become a line representing the number of line segments passing through the respective bin. The visual output of this procedure would be identically with a visualization where the raw data is displayed using line segments to connect the time series data. Although this



(a) (b) (c)

**Figure 3.9:** Comparison of the granularity of the visualization. (a) $64 \times 64$ bins per map; (b) $128 \times 128$ bins per map; (c) $256 \times 256$ bins per map.

example is illusory, it demonstrates that the granularity of the output is highly dependent on the number of bins used.

Therefore, the decision is left to the user, allowing him/her to interactively alter the resolution of the maps, which makes the approach scalable. One can, for instance, advance the level of details in the visualization, by incrementing the resolution of the bin maps. On the other hand, the user can reduce the number of bins per map, to shorten rendering time and enhance interaction. In figure 3.9 different resolutions are used for the bin maps, which affects the granularity of the visualization. Several highlighted time series (red), which are discriminable in Fig. (c) merge in the visualization in the figures (b) and (a). The smooth and continuous appearance of the curves represented in Fig. (c) highly suffers when the bin map resolution is reduced (e. g., see Fig. (a)). Per default, a resolution of $256 \times 256$ bins per map are used.

### Zooming into the Visualization and Re-Binning:

One may have the idea to re-bin the data when the user vertically zooms into the visual representation. In this case, only the time series having values inside the visible area of the CurveView (indicated by the red rectangle in Fig. 3.10 (a)) would be aggregate to keep the granularity of the visualization constant. However, as one can see in figure 3.10, then time series crossing the visible area would be lost in the visualization (e. g., see the green line segment). Hence, the data is not re-binned in the proposed approach, and the bin maps can be re-used while zooming and panning.

On the other hand, when depicting the binned data representation, certain areas in the bin map can be skipped, indicated by the gray squares in figure 3.10 (a). This is due to the fact, that these regions represent time series, where the respective data values are both outside the visible area, but do not cross it. A possible advancement for future research is, to aggregate the values outside the zoomed area applying a different (lower) resolution, and to increase the number of bins used for the visible area.

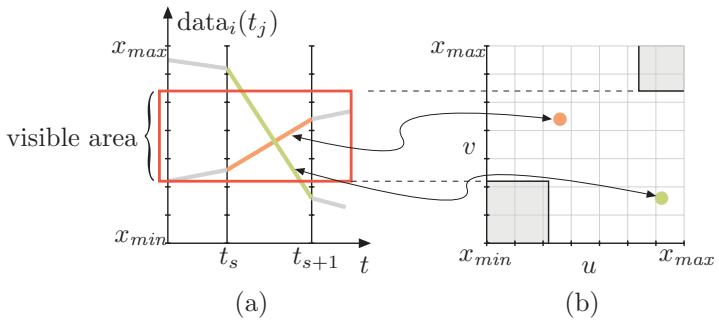


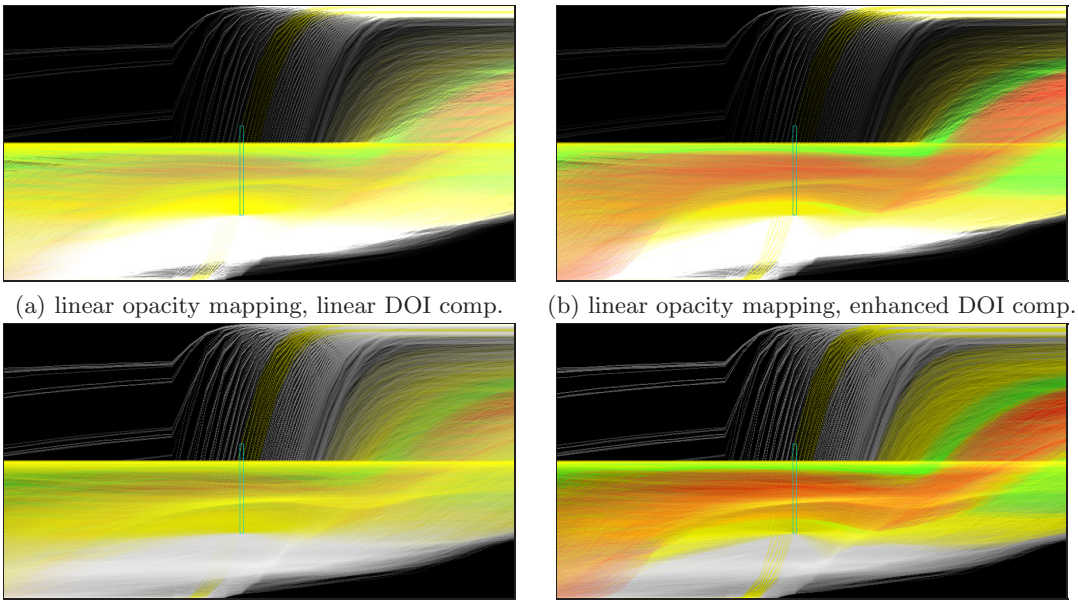

**Figure 3.10:** (a) The user has zoomed into the CurveView, the visible area is indicated by a red rectangle. When aggregating only the zoomed area using 2D bin maps, time series crossing this area would be lost (e. g., the green line segment). (b) When depicting the binned data, the gray squares can be skipped, as the respective line segments are not visible.

### 3.4.2 Comparing Different Mappings in Focus+Context Visualization

In figure 3.11 different mappings in the CurveView are illustrated, showing the sample feature specification previously described (see Fig. 3.8 in Sec. 3.3.2). Thereby, a *linear* opacity transfer function is applied in figure 3.11 (a) and (b), which maps the number of time series, running through a pixel, to the respective opacity (intensity) value. Thereby, a steep slope is used in the figures (a) and (b), to emphasize the main portion of the time series, and to make the color coding (feature visualization) visible. However, this has the drawback, that visual structures (due to intensity) in dense areas of the visualization are hardly discriminable. When scaling down the linear mapping the trends are revealed, while the features disappear. On the other hand, a *logarithmic* opacity mapping is used in figure 3.11 (c) and (d). Here, the outliers are enhanced, while preserving the general data trends in the visualization. However, due to the non-linear mapping, one can not intuitively estimate the number of overlapping time series from the color intensities. For this purpose, a linear mapping is recommendable.

In figure 3.11 (a) and (c) a *linear* function is used (i. e., $\gamma = 1$ in Eq. 3.8), to calculate the normalized DOI values, applied in the importance-driven color coding. Accordingly, the red and green time series (i. e., the features) are hardly discriminable in the visualization. On the other hand, a *square-root* function is applied (i. e., $\gamma = 0.5$ in Eq. 3.8) in the CurveViews shown in figure (b) and (d). Therefore, the DOI values (and the features) are represented in an enhanced manner, i. e., the red and green time series are revealed. However, this does not correspond to the real amount of important time series in the visualization, which is better represented using a linear mapping.

These examples illustrate, that it is quite difficult to convey all the important information (features, patterns, data trends, etc.) within one static image. Different specifications emphasize different aspects of the data. Therefore, *interaction* is a key issue in visual analytics, which is also true when analyzing time series data. The switching between the different combinations of mappings, and the interactive modification of the specifications



(a) linear opacity mapping, linear DOI comp.

(b) linear opacity mapping, enhanced DOI comp.

(c) log. opacity mapping, linear DOI comp.

(d) log. opacity mapping, enhanced DOI comp.

**Figure 3.11:** Comparison of different opacity transfer functions and DOI enhancement: In the upper row a linear mapping of density to opacity values is applied, while a logaritmic mapping is applied in the bottom row. In the left column the normalized DOI information is representation, while this information is shown in an enhanced manner (using a square root function) in the right column.

affecting the visualization, allow the user to gain insight into the nature of the data (compare to Sec. 1.1.1 and Sec. 2.1.1).

# 4 Brushing Time-Series Data

## Interactive Specification of Time-Dependent Features

The following chapter proposes several smooth brushing approaches: *Time step brushes* (see Sec. 4.2) where time series are selected that run through a specified interval at a certain time step; and *similarity-based* approaches where time series are classified according to their similarity to a user-defined pattern, which is directly sketched in the CurveView (see Sec. 4.3). In section 4.4 different kinds of these brushes are proposed, where the similarity is measured based on the gradients of the time series and the brush. The related state-of-the-art of techniques allowing the specification of time-dependent features was described previously in section 2.2.3.

## 4.1 Interactive Feature Specification

In the context of *visual analytics* [83, 82] (see Sec. 2.1.1) and Keim's *visual analytics mantra* [40] (see Sec. 2.1.1), *interaction* and *human factors* are of major importance when specifying features via brushing. These factors are, for instance, human intuition, domain knowledge of the expert using the application, as well as the excellent ability to detect visual structures, which appear during brushing or when altering the specifications of the visualization (e. g., the parameters described in Sec. 3.4, such as linear/logarithmic mapping, scale factor, constant offset, number of bins). In this context, feature specification is an interactive and *iterative* process where the user searches for unknown and interesting patterns in the data, and furthermore aims to extract these features using brushing techniques.

As it was described in section 2.1.2, the *SimVis* system [15, 21] uses *smooth brushes* [17] for fuzzy classification [96] of the data items displayed in a view. The brushes are organized and combined using fuzzy-logic operations in a hierarchical *feature definition language* (FDL) tree-like structure [15]. Via *brushing* [94, 5] the user actively specifies his/her interest on a subset of the displayed data—called manual *feature specification* (compare to the work done by Doleisch et al. [18])—by directly marking the data items on the display screen. As a result, fractional *degree-of-interest* (DOI) values (compare to Furnas [27]), which are assigned to each data entry in a SimVis view, from the interval $[0, 1]$ are altered. DOI values also build the basis for *focus+context visualization* in SimVis (compare to the work of Hauser [30] and Doleisch et al. [16, 17, etc.]), where data items with maximal user interest—the focus portion of the data with DOI values equal to 1—are represented in an

enhanced manner. The rest of the data, where the DOI is 0 (context portion of the data), is depicted in a reduced style for orientation. Thereby, a continuous border region, where the DOI gradually changes between one (i. e., 100% user interest) and zero (no interest), is assumed between focus and context.

## 4.2 Smooth Time Step Brushes

In the context of smooth brushing, a *time step brush* represents a (smooth) vertical selection in the CurveView on a certain time step (e. g., see Fig 4.1 or Fig. 4.2 where multiple brushes are logically combined). A single brush is specified by a time step $t_s$ (in the time-dependent data set $\mathcal{S}$ in Sec. 1.2.1) and a tuple of scalar thresholds $(b_1, b_2, b_3, b_4)$ in the codomain of the data $[x_{min}, x_{max}]$. When evaluating the brush, the data value $x_{i,s} = \text{data}_i(t_s)$ of each time series $\mathcal{D}_i \in \mathcal{S}$ (given at the time step of the brush $t_s$) is considered and the resulting fractional *degree-of-interest* (DOI) value is assigned to all data items of the respective time series $\mathcal{D}_i$. The calculation of the DOI values is illustrated in figure 4.1, and can be defined as a one-dimensional (trapezoidal) function (compare to the work of Doleisch [15] or Hauser [30]).

$$\text{doi}_{i,j}(x_{i,s}, b_1, b_2, b_3, b_4) = \begin{cases} 0 & \text{if } x_{i,s} \leq b_1 \vee x_{i,s} \geq b_4 \quad \text{(context)} \\ \frac{x_{i,s}-b_1}{b_2-b_1} & \text{if } b_1 < x_{i,s} < b_2 \\ 1 & \text{if } b_2 \leq x_{i,s} \leq b_3 \quad \text{(focus)} \\ \frac{b_4-x_{i,s}}{b_4-b_3} & \text{if } b_3 < x_{i,s} < b_4 \end{cases}, \tag{4.1}$$

were $x_{min} \leq b_1 \leq b_2 \leq b_3 \leq b_4 \leq x_{max}$. The data values $x_{i,s}$ of the time series that fall into the interval $[b_2, b_3]$, which specifies the *inner region* of the brush (orange rectangle in Fig. 4.1 (a)), receive a maximal DOI value ($1 \leftrightarrow 100\%$ user interest) and are considered to be in *focus*. Time-dependent data values at time step $t_s$ that lie outside of the boundaries of the time step brush (i. e., $x_{i,s} \leq b_1$ or $x_{i,s} \geq b_4$) receive a DOI value of zero (no user interest) from the function and are considered to belong to the *context* portion of the data (see Fig. 4.1 (b)). Data values $x_{i,s}$ belonging to the interval $]b_1, b_2[$ or $]b_3, b_4[$, which define the *outer regions* of the smooth time step brush (light orange rectangles in Fig. 4.1 (a)), get a DOI value from the interval $]0, 1[$ from the function in (4.1). These regions are the boundaries between focus and context, where a gradually increasing or decreasing degree-of-interest (DOI) is assumed (see the trapezoidal shape in Fig. 4.1 (b)).

As it was already mentioned, the resulting fractional DOI values from (4.1) are then assigned to all data items $x_{i,j} \in \mathcal{D}_i$ in the SimVis view, where $j = 1, \ldots, N$ and $N$ is the number of discrete time steps in the data set $\mathcal{S}$ (and the time series). The CurveView allows the creation of an arbitrary number of time step brushes, which are logically combined in the hierarchical *feature definition language* (FDL) tree in SimVis [15], both described in the following. The (smooth) selections of the brushes are reflected immediately in all linked views by the propagation of the DOI values along the FDL tree and the depiction

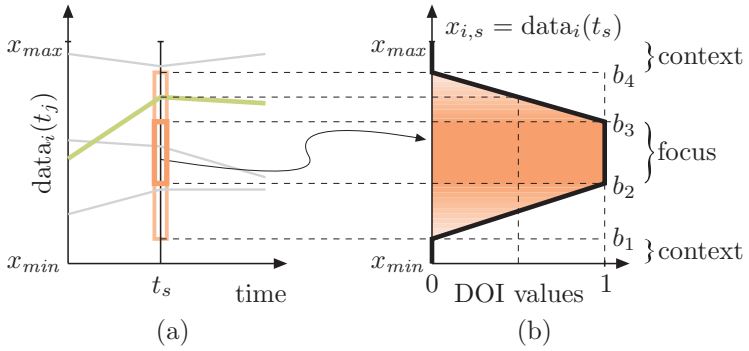



**Figure 4.1:** Brushing time series data using a *time step brush*: (a) The time-dependent data values $x_{i,j} = \text{data}_i(t_j)$ of the time series $D_i$ are given at discrete time steps $t_j$ and are interconnected with gray polylines. At a certain time step $t_s$ the data set is selected using a smooth time step brush (orange rectangle). The illustration in (b) is adapted from Doleisch [15].

of the data and the associated DOI information using importance-driven color coding (*focus+context visualization* [30]) as it was described in section 3.3.2.

Other approaches known from literature, similar to this kind of brushing were briefly described in section 2.2.3: *Timeboxes* are rectangular querying widgets in the TimeSearcher application [36, 35] which are used to select all time series running through the rectangular query box drawn in a view. Another brushing approach is *line brushing* [49], simple line segments connecting two arbitrary points in a view, which are used to select families of function graphs that intersect with the line brush. However, both applications use binary classification to discriminate whether a time series is selected or not. In contrast to that, time step brushes in SimVis apply fuzzy classification [96]. All the mentioned applications have in common that multiple brushes can be combined using logical operations (e. g., AND, OR, NOT). A more in depth description is provided in section 4.2.2.

### 4.2.1 Intuitive Creation and Modification of Time Step Brushes

The creation of a time step brush in the CurveView is a straight-forward action performed by the user: By clicking with the left mouse button on the desired starting position in the view and dragging the mouse pointer to the end position of the brush, the inner region $(b_2, b_3)$ and the time step $t_s$ are specified, which denote the parameters used in the DOI evaluation in (4.1). Initially, the smooth outer regions are limited to the inner region of the brush, i. e., $b_1 = b_2$ and $b_3 = b_4$. The process of creating a time step brush is quite similar to the construction of simple geometric primitives (e. g., rectangle, straight line) in commonly used drawing applications (e. g., Adobe Illustrator[1]) or the creation of Timeboxes in TimeSearcher [36], for instance.

The smooth bounds of the outer regions of the time step brush can be altered by the

[1] http://www.adobe.com/

user by clicking on the desired upper or lower edge and dragging it while the "Shift"-key is pressed. All edges of the brush $(b_1, b_2, b_3, b_4)$ are sensitive to the mouse pointer—the cursor shape changes when an edge is touched by the mouse pointer—and can be easily modified with click-and-drag operations. In addition to that, the parameters of the brush can be altered numerically using a corresponding pop-up dialogue, which can be selected from a pop-up menu when the user right-clicks on the brush. Furthermore, the whole component can be moved to a new location in the view by clicking inside the visual representation of the brush and dragging it.

While creating and modifying a time step brush the user gets continuous visual feedback from the SimVis system on the features specified by one or more brushes. The smooth selections are visually highlighted in all (linked) views using focus+context visualization. Based on the visual response, the brushes can be further refined in order to explore the interrelations of the data items, or the temporal evolution of time series intersecting with the brush.

### 4.2.2 Combining multiple Time Step Brushes in the FDL Tree

As it was already described in section 2.1.2, the brushes (and views) in SimVis [15, 21] are organized in a hierarchical *feature definition language* (FDL) tree-like structure. Here, multiple smooth brushes are interpreted as *fuzzy sets* [96] and can be combined using fuzzy-logic operations (AND, OR, NOT) realized as T-norms and T-conorms (compare to Klement et al. [47]). This allows the specification of complex features using multiple smooth brushes in multiple views. The CurveView also allows the combination of multiple time step brushes (and similarity brushes described in Sec.4.3). Three types can be described (see Fig. 4.2):

**Smooth AND-brushes:** All brushes assigned to this type within a view—each SimVis view which allows brushing, is represented as a feature component node in the FDL tree (see Fig. 4.2 (left))—are combined using an implicit (fuzzy) AND-operation. In other words, time series which are selected smoothly in a view have to intersect with all respective *AND-brushes*, depicted yellow in figure 4.2 (right).

**Smooth OR-brushes:** This type of time step brush is realized in the FDL tree with a fuzzy OR-operation (see Fig. 4.2 (left)). Time series that pass through a time step *OR-brush* (depicted green in Fig. 4.2) receive a DOI value $> 0$ if the selection is not restricted by other brushes of a different type (e. g., AND, NOT).

**Smooth NOT-brushes:** Finally, the user can also exclude data items from the selection using *NOT-brushes*, which are first combined in the FDL tree by a fuzzy OR-operation and then excluded from the fuzzy set using a NOT-AND operation (see Fig. 4.2 (left)). The respective brushes in the CurveView are depicted pink in figure 4.2 (right).

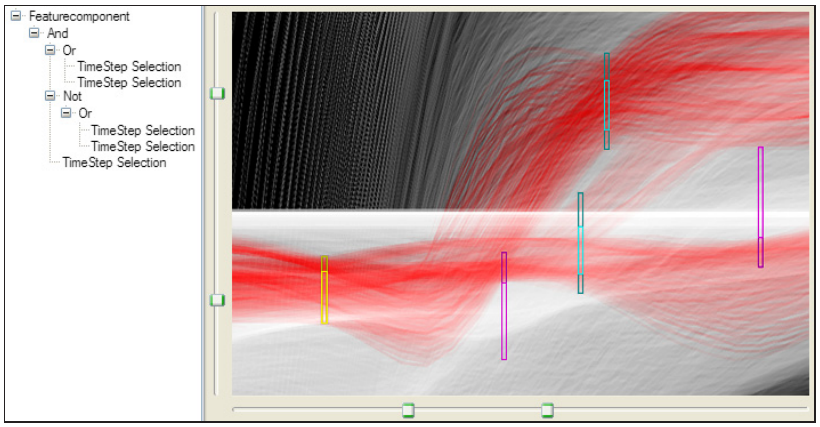



**Figure 4.2:** Illustration of the logical combination of multiple types of *time step brushes* in the FDL tree: OR-brushes are depicted green, AND-brushes yellow, and NOT-brushes pink. For simplicity only a part of the FDL tree is illustrated (left), namely the child nodes of the *feature component node*, which represents the CurveView (right). Features are amplified using a nonlinear DOI computation ($\gamma \approx 0.4$ in Eq. 3.8).

## 4.3 Similarity-based Smooth Brushing of Time-Dependent Features

The idea of the proposed *similarity-based brushing* approach is that the user sketches a time-dependent pattern directly in the CurveView by specifying an arbitrary number of control points. The points are then interconnected by a polyline which represents the shape of the *similarity brush* used for querying (see Fig 4.3). Time series are classified smoothly (compare to *fuzzy classification* [96]) according to their similarity to the user-defined reference pattern. The brush is evaluated based on a similarity measure between the pattern and the respective time series, which results in a fuzzy classification of the time series. This means, that an adequate degree-of-interest (DOI) value is assigned to the data items of the time series, according to the similarity measurement. Based on the respective features visualization during the analysis session, the user can further extend or refine the similarity brush.

Humans have a subjective sensation of whether two patterns (e. g., curves or time series) are similar or not. Three examples are illustrated in figure 4.4: In (a) the two sequences are very much alike, although they are not equal. In figure (b) a constant (vertical) offset is added to one of the pattern from (a). In (c) the two patterns are similar at many spots, but there is a single outlier in the orange pattern. One could start arguing, whether the figures (b) and (c) show similar patterns or not. However, in the context of *visual analytics* and brushing, certain metrics have to be considered, which quantify the similarity between two patterns and, therefore, build the basis for the approach proposed later in section 4.4.

In the following a similarity brush is described in a general manner (see Sec. 4.3.1), moreover, how such a brush can be created and modified in the CurveView (see Sec. 4.3.2).

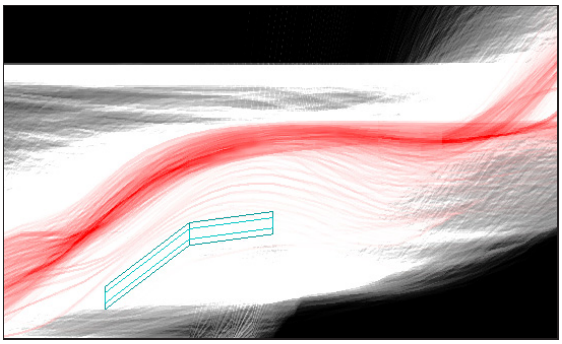


**Figure 4.3:** Similarity-based brushing example: Time series are highlighted in the CurveView, according to their alikeness to a *similarity brush*. The visual representation of the brush is a polyline. Two thresholds, which are used for similarity-based DOI computation of the time series, are mapped to the vertical dimension (height) of the inner and outer region of the similarity brush.

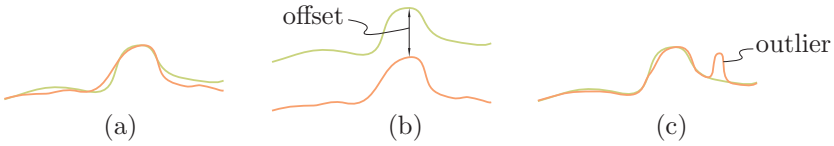



**Figure 4.4:** Are these patterns similar? (a) two patterns; (b) a constant offset is added to one of the patterns; (c) the orange pattern contains an outlier.

Then the issue of similarity measurement between time series and a brush is considered in section 4.3.3, and how this can be related to smooth brushing of time-dependent features.

### 4.3.1 Defining a Similarity Brush

A *similarity brush* is defined by an arbitrary number of *control points*, which are explicitly specified by the user, and by two thresholds. Such a brush can be created, for instance, by clicking with the mouse on the desired screen positions of the control points (in the CurveView), or by numerically altering the point values. Thus, a similarity brush $\mathcal{SB}_k$ is defined by a sequence of $P$ tuples of data values $\hat{x}_{k,l}$ and time values $\hat{t}_{k,l}$, i. e.,

$$\mathcal{SB}_k = \{(\hat{x}_{k,1}, \hat{t}_{k,1}), (\hat{x}_{k,2}, \hat{t}_{k,2}), \ldots, (\hat{x}_{k,P}, \hat{t}_{k,P})\}. \tag{4.2}$$

The data values $\hat{x}_{k,l}$ are specified in the data domain of the time-dependent attribute (i. e., the dimension assigned to the CurveView) and the corresponding time values $\hat{t}_{k,l}$ are defined in the associated time domain. Moreover, each $\hat{t}_{k,l}$ has to be less than $\hat{t}_{k,l+1}$, to avoid invalid patterns. To simplify the evaluation of the brushes the time values $\hat{t}_{k,l}$ in (4.2) are restricted to the discrete time steps $t_s$ of the time-dependent data set $\mathcal{S}$ (see Sec. 1.2.1). When sampling the similarity brush at discrete brush time steps $\tilde{t}_j$—these are normally equivalent to the time steps $t_s$ of the time series—the corresponding brush values are denoted as $\tilde{x}_{k,j} = \text{brush}_k(\tilde{t}_j)$.

The visual representation of a similarity brush in a CurveView is a sequence of line segments (i. e., polyline) connecting the control points, as depicted in the brushing example in figure 4.3. Two thresholds used for similarity-based smooth brushing of time series—this will be described and formalized later in this section (see Eq. 4.5)—are mapped to the height of the brush. As it is shown in figure 4.3, the first threshold is mapped to the inner region of the brush and the second threshold to the (upper and lower) outer regions.

### 4.3.2 Creating and Modifying a Similarity Brush

Also the creation of a similarity brush is a straight-forward action performed by the user (compare to time step brushes): By clicking with the left mouse button on the desired position of a control point in the view and dragging the mouse pointer to the next position, the first line segment of the brush is specified. In a special *creation mode* the brush can be extended interactively, adding further control points to it. Thereby, the new point continuously follows the current mouse pointer until the user clicks down. Then, the new position of the point is specified. Pressing the "Enter" or "Escape"-key the creation mode can be left. Then, single control points can be modified by clicking and dragging them. Thereby, the points are always interconnected from left to right according to their horizontal position, to avoid invalid patterns (as defined in Sec. 4.3.1). While modifying or creating a brush, the corresponding feature is continuously evaluated (i. e., the DOI values), which affects the FDL tree and the visualization in the linked SimVis views. Therefore, the user gets immediate visual response to the feature currently specified.

Both thresholds of the brush can be altered by the user by clicking on the desired upper or lower edge at an arbitrary control point and dragging it while the "Shift"-key is pressed (compare to time step brushes). In addition to that, the thresholds and single control points can be altered numerically using a corresponding pop-up dialogue, which can be selected from a pop-up menu when the user right-clicks on the brush. Furthermore, the whole component can be moved to a new location in the view by clicking inside the visual representation of the brush and dragging it.

### 4.3.3 Similarity Measurements and Smooth Brushing on Time Series Data

To measure whether two sequences of values (e. g., subsequences in time series) are similar or not, a scalar value, called the *distance*, is used to quantify the amount of similarity or dissimilarity. Distance metrics play a central role in many applications of data mining [44] (of time series), such as clustering [95], time series classification [71], anomaly detection [43] or temporal pattern matching [9, 13]. Moreover, they build the basis for *similarity-based querying* of time series where the distance of a subsequence of time-dependent data values to a certain reference sequence is computed—the reference sequence is commonly denoted as *signature*, *pattern* or *template*. Then, the distance value is compared to a *tolerance threshold*, specified by the user, to discriminate whether the time series is considered to be similar or not (see Hetland [34] for a survey, or Andrienko and Andrienko [4, Chap. 4.6]).

Several metrics exist for quantifying similarity between (sub-)sequences of (e. g., time-dependent) values. In many cases it is beneficial to apply *data transformations* on the raw data values before *similarity measurement* (compare to Keogh and Kasetty [42]), which will both be described in general in the following. After that, the remainder of this section deals with the issue of *similarity-based classification* of time series—in SimVis fuzzy classification is incorporated as smooth brushing using fractional degree-of-interest (DOI) values.

### Distance Measurements between two Sequences of Values

In the following the general basics related to distance measurements will be described. Note that, however, the proposed approach computes similarity based on relative (derived) attributes, such as the distance of gradients, slopes or angles, rather than on absolute data values (see Sec. 4.4).

According to the work of Andrienko and Andrienko [4, Chap. 4.6], some commonly used *distance measurements* in literature can be generalized in the family of *Minkowski metrics*, also called $\mathcal{L}_P$ norms. These metrics quantify the distance between two sequences of an equal number of $N$ values (e. g., subsequences in time series), namely $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$ and $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$, and are defined as

$$\mathrm{dist}_{\mathcal{L}_P}(\mathbf{x}, \mathbf{y}) = \Big[\sum_{i=1}^{N} |x_i - y_i|^P\Big]^{\frac{1}{P}}. \tag{4.3}$$

In the case of $P = 1$, equation 4.3 represents the so called *Manhattan* or *city block distance*, where the respective absolute differences of the values are summed up. The *Euclidean distance* is represented in (4.3), if $P = 2$. According to Keogh and Kasetty [42], this metric has a very low error rate and is used in the bigger part of published work on time series data mining[2] because of its formal properties.

Another method for computing the distance between the sequences $\mathbf{x}$ and $\mathbf{y}$ is to consider only the *maximal absolute difference* between the particular values, i. e.,

$$\mathrm{dist}_{max}(\mathbf{x}, \mathbf{y}) = \max_{i=1,\ldots,N} |x_i - y_i| \,. \tag{4.4}$$

Thereby, a sequence is considered to be dissimilar if only one absolute difference between the values of the time series and the pattern is greater than the tolerance threshold. Hence, this approach is very sensitive to time series with outliers concerning similarity, as illustrated in figure 4.4 (c). On the other hand, using a Minkowski metric, time series can be considered to be similar if most of their values are very much related to the reference pattern and only a few variations (e. g., outliers, noise) exist, as depicted in figure 4.4 (a) and (c).

[2] According to a survey on time series data mining done by Keogh and Kasetty in 2002 [42], about 80% of the accounted work uses Euclidean distance.

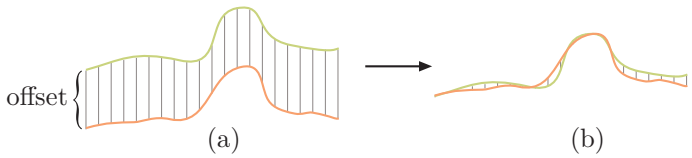



**Figure 4.5:** Value Transformations for similarity measurement: (a) If a constant offset is added to the values of the time series (or pattern) the calculated distance (grey lines) can become meaningless for the purpose of (subjective) similarity measurement. (b) The respective mean values are subtracted from the values of the time series and the pattern, called value transformation. Then the distance (similarity) is calculated on the resulting values. The illustration is adapted from Keogh and Kasetty [42].

### Value Transformations for Similarity-based Brushing

When computing the similarity (distance) on the original values of the reference pattern and the corresponding subsequence in the time series (as it was just described), this can have some unwanted drawbacks: If, for instance, a constant scalar offset is added to the values of the time series (or the pattern), which is called *offset translation*, the resulting distance between pattern and time series increases in proportion to the length of the gray lines in figure 4.5 (a). According to Keogh and Kasetty [42], the distance can then become meaningless for the purpose of (subjective) similarity measurement, as it overrates the distance perceived by an observer. It is therefore advantageous to *transform* the raw data values into a normalized representation and to apply distance measurement on the normalized values instead of the original ones (see Fig. 4.5 (b)). Such a value transformation can be done by subtracting the respective mean value from both the values in the pattern and in the time series subsequence before computing their distance, for example.

There are also other issues when evaluating the similarity of the raw data values, such as *magnitude scaling*, *linear trends removal* or *noise reduction* (see Keogh [41]). The approach for similarity-based brushing presented in this thesis, however, focuses on the offset translation problem illustrated in figure 4.5, as this is relatively obvious to the user when searching for time series which are (subjectively) similar to a user-defined pattern. Therefore, the distance measurement is applied on derived attributes of the time series data (e. g., gradients or the slope between the values given at certain time steps), which are robust against offset translation of the raw values. This approach will be described later in section 4.4.

### Fuzzy Time Series Classification according to their Similarity

For similarity-based querying a scalar *tolerance threshold* specified by the user, is commonly used to discriminate whether a subsequence of a time series is similar to a certain (user-defined) reference sequence (e. g., called signature, pattern or template). If the particular (calculated) distance value is equal or less than the threshold, the time series is classified as similar, otherwise it is considered to be dissimilar, which corresponds to a

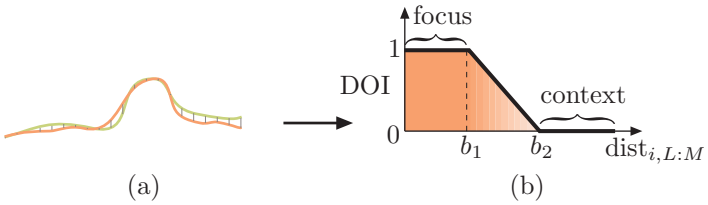



**Figure 4.6:** Fuzzy classification of a time series according to its distance to a user-defined *similarity brush*: (a) The distance $\text{dist}_{i,L:M}$ between the time series (green curve) and the specified reference pattern (orange curve) is evaluated. (b) A fractional degree-of-interest (DOI) value from the interval $[0, 1]$ is assigned to the time series, according to its distance $\text{dist}_{i,L:M}$.

binary classification [34].

As SimVis, however, uses *smooth brushing* [17] (see Sec. 2.1.2), two thresholds, namely $b_1$ and $b_2$, are applied for the similarity-based *fuzzy classification* [96] of time series in a data set $\mathcal{S}$. When evaluating a smooth similarity brush, a scalar *degree-of-interest* (DOI) value [27] from the unit interval is assigned to the items of a certain time series $\mathcal{D}_i \in \mathcal{S}$, according to its (dis-)similarity to the reference pattern (the shape of the brush). The DOI assignment is illustrated in figure 4.6 and can be formalized as

$$\text{doi\_similarity}_i(\text{dist}_{i,L:M}, b_1, b_2) = \begin{cases} 1 & \text{if } 0 \leq \text{dist}_{i,L:M} \leq b_1 \quad \text{(focus)} \\ \frac{b_2 - \text{dist}_{i,L:M}}{b_2 - b_1} & \text{if } b_1 < \text{dist}_{i,L:M} < b_2 \\ 0 & \text{if } b_2 \leq \text{dist}_{i,L:M} \quad \text{(context)} \end{cases} \tag{4.5}$$

where $\text{dist}_{i,L:M}$ denotes the computed distance between a subsequence of time-dependent data values in $\mathcal{D}_i$, and the corresponding values in the reference template. Note that only the time-dependent values of the time series $\mathcal{D}_i$, which are given between the time steps $t_M$ and $t_N$ (inclusive), are considered when evaluating the distance. Thereby, $t_M$ and $t_N$ denote the first and last time step in $\mathcal{D}_i$ where the similarity brush is specified, respectively. An example of brushing time series according to similarity to a user-defined similarity brush is depicted in figure 4.3. The two thresholds are mapped to the height of the inner and outer region of the similarity brush.

In equation 4.5 and figure 4.6, time series having a distance $\text{dist}_{i,L:M}$, which is less or equal to the threshold $b_1$, are considered belonging to the *focus* portion of the data (inner region of brush in Fig. 4.3). Accordingly, the associated data items in $\mathcal{D}_i$ receive a maximal DOI value of one. If the distance is equal or greater than the second threshold $b_2$ the elements in the time series receive a DOI value of zero (*context* portion of the data). Time series with a distance inside of the interval $]b_1, b_2[$ get a DOI value from $]0, 1[$ in (4.5). Thereby, the range between $b_1$ and $b_2$ (outer region of the brush in Fig. 4.3) is considered to be the boundary between focus and context, where a gradually increasing DOI value applied, as illustrated in figure 4.6 (also compare to smooth time step brushes in Sec. 4.2).

In the similarity based brushing approach presented in this thesis, the sum of absolute differences ($P = 1$ in Eq. 4.3) is applied to quantifying similarity. This forms a useful

compromise between a relatively low error rate and a distance measurement incorporating all values of the pattern and the subsequence of the time series. Moreover, the linear behavior of the city block distance fits very well with the gradually increasing DOI values at the boundary between the focus and context portion of the data (see Eq. 4.5 and Fig. 4.6). As mentioned before, the distance is computed on derived attributes of the time series (e. g., data gradients or slopes) to deal with the offset translation problem illustrated in figure 4.5.

## 4.4 Types of Gradient-Based Smooth Similarity Brushes for Time-Dependent Feature Specification

The proposed *gradient-based* smooth similarity brushing approach aims to select time series similar to a user-defined pattern, which is directly sketched in the CurveView. Thereby, similarity measurement is based on the respective gradients of the brush and the time series. The fractional degree-of-interest (DOI) values assigned to each data item of a time series are altered according to the similarity to the reference pattern. The resulting time-dependent features are depicted in real-time (in multiple linked views) using focus+context visualization (see Chap. 3). As the time series are not necessarily sampled at regular time intervals (i. e., at evenly spaced time steps), the respective differences are weighted properly before adding them up. In the context of smooth brushing, the resulting distance is then compared to two thresholds discriminating focus from context. Thereby, a DOI value for the time series is computed (see Eq. 4.5).

The approach presented in this section is inspired by brushing techniques, such as *angular query widgets* [36, 35], *angular brushig* [32] or *pattern search* in TimeSearcher [9] (see Sec. 2.2.3), where items are brushed based on ranges of differences, rather than on absolute values. Three types of similarity-based brushes were integrated, such as *gradient-Distance-sum* (GDS), *angular-distance-sum* (ADS) and *point-sampled-slope* (PSS) brushes, described later. In the following *gradient estimation*, and *weightin*g in order to cope with the unevenly spaced time steps are described.

### 4.4.1 Calculating Gradients and Weights for Similarity-Based Brushing

As done in previous work (see Doleisch et al. [15, 16]), smooth brushing is applied on the gradients ($x_i'(t) = \frac{\mathrm{d}x_i}{\mathrm{d}t}$) of a time-dependent attribute $x_i = f_i(t)$, in order to detect temporal patterns of changes. This enables the user to brush similar time-series, which exhibit, for instance, the offset translation problem previously described in section 4.3. However, up to now the derived attributes (e. g., gradients) could only be brushed in a separate attribute view, e. g., using a rectangular smooth brush in a scatterplot or histogram view. The presented gradient-based similarity brushing approach, allows the user to implicitly brush time series gradients in a very intuitive way by specifying complex patterns (e. g., an arbitrary polyline) directly in the CurveView, where the time series data is visualized.

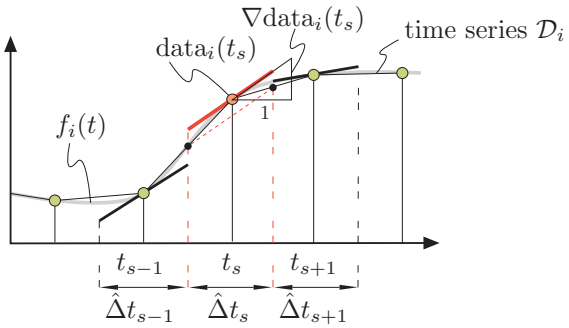



**Figure 4.7:** Calculating gradients and weights on time series data: An attribute being a continuous function of time (grey curve $f_i(t)$) is sampled at discrete time steps, which builds the time series $\mathcal{D}_i$. The *gradient* $\text{data}_i(t_s)$ (red line) at a certain time step $t_s$ in $\mathcal{D}_i$ can be calculated as a weighted sum of the three successive time steps, namely $t_{s-1}$, $t_s$, and $t_{s+1}$. Each gradient at a time step $t_s$ is considered to be valid (and constant) for a certain *time interval* $\hat{\Delta}t_s$, which is also the weight of the associated gradient.

**Gradient Estimation:** As the time-dependent data values $x_{i,j} = \text{data}_i(t_j)$ of the time series $\mathcal{D}_i$ are sampled at discrete time steps $t_j$ over time (see Eq. 1.2), three types of *gradient estimation* are commonly applied as linear filter functions[3]: forward, backward, or central differences. Because of its formal properties (e. g., relatively low error rate, data smoothing), central differences are used in SimVis, illustrated in figure 4.7, which can be calculated as

$$\nabla\text{data}_i(t_s) = \frac{1}{2}\Big(\underbrace{\frac{\text{data}_i(t_{s+1}) - \text{data}_i(t_s)}{t_{s+1} - t_s}}_{\text{right slope}} + \underbrace{\frac{\text{data}_i(t_s) - \text{data}_i(t_{s-1})}{t_s - t_{s-1}}}_{\text{left slope}}\Big) \tag{4.6}$$

Roughly speaking, the gradient $\nabla\text{data}_i()$ at a certain time step $t_s$ in the time series is calculated as the mean value of the right and the left slope (compare to Doleisch [15]). The right slope is between the data values $\text{data}_i(t_{s+1})$ and $\text{data}_i(t_s)$ given at the successive time steps $t_{s+1}$ and $t_s$, respectively, and the left slope is between the data values given at the time steps $t_s$ and $t_{s-1}$ (see Fig. 4.7). This compensates for unevenly spaced time steps in the time series data, and performs data smoothing due to the width of the filter, i. e., multiple data values are incorporated into the gradient estimation.

**Calculating Weights:** The estimated gradient $\nabla\text{data}_i(t_s)$ is considered to be valid (and constant) for a certain time interval $\hat{\Delta}t_s$ on the time axis, illustrated in figure 4.7. The interval $\hat{\Delta}t_s$ is also the weight of the gradient, which will be used later when calculating similarity, and can be computed as

$$\text{weight}(t_s) = \hat{\Delta}t_s = \frac{1}{2}(t_{s+1} - t_{s-1}). \tag{4.7}$$

[3] A *linear filter function* is a weighted sum of discrete data samples, where the data and a *filter kernel* of certain size (containing weights) are incorporated, i. e., the data samples within the filter region are multiplied with the associated weight and summed up.

Here $t_{s+1}$ and $t_{s-1}$ denote the time values of the time steps, which are adjacent to $t_s$. If the time steps of the time series $\mathcal{D}_i$ are evenly spaced, equation 4.6 represents the commonly used form of central differences ($\frac{\text{data}_i(t_{s+1})-\text{data}_i(t_{s-1})}{2(t_{s+1}-t_{s-1})}$) and all weights in (4.7) are equal.

### 4.4.2 Gradient-Distance-Sum (GDS) Smooth Similarity Brushes

The first type of smooth similarity brushes applied in the brushing approach presented here, is the *gradient-distance-sum* (GDS) similarity brush, which is illustrated in figure 4.8. As implied by its name, the *absolute differences* between the gradients of a GDS similarity brush $\mathcal{GS}_k$ and a time series $\mathcal{D}_i$ are weighted and summed up. This corresponds to a slightly modified version of the Manhattan or city block distance ($P = 1$ in Eq. 4.3), where every gradient difference is weighted in order to "recreate" the (offset) transformed (and smoothed) time series data values[4]. The metric is used due to the fact that the sum of absolute distances, in contrast to other metrics (e. g., Euclidean Distance), has a linear behavior. This corresponds well with the linearly increasing/decreasing DOI values at the boundary between the focus and context portion of the data (see Eq. 4.5 and Fig. 4.6).

**Evaluating Similarity:** The calculation of the distance (dissimilarity) between the subsequence of a time series $\mathcal{D}_i$ and a GDS similarity brush $\mathcal{GS}_k$ can be formalized as

$$\text{dist}_{grad}(\mathcal{D}_i, \mathcal{GS}_k) = \sum_{s=L}^{M} \hat{\Delta}t_s \cdot \underbrace{\left|\nabla\text{data}_i(t_s) - \nabla\text{brush}_k(t_s)\right|}_{\text{dist}_{grad}(\mathcal{D}_i,\mathcal{GS}_k,t_s)}. \tag{4.8}$$

The gradient of the brush $\nabla\text{brush}_k(t_s)$ and of the respective time series $\nabla\text{data}_i(t_s)$ are both calculated at a certain time step $t_s \in \mathcal{D}_i$ using central differences (compare to Eq. 4.6). The two gradients are considered to be representative (and constant) for a certain time interval $\hat{\Delta}t_s$, which is associated with a time step $t_s$ of the time series subsequence (see Fig. 4.8). The absolute difference between both gradients is denoted as $\text{dist}_{grad}(\mathcal{D}_i, \mathcal{GS}_k, t_s)$ in (4.8) and is also illustrated in figure 4.8 (b). As the time intervals $\hat{\Delta}t_s$ need not be of equal length (i. e., the data is sampled at irregularly space time steps), each gradient distance is weighted with the duration of the associated time interval ($\text{weight}(t_s) = \hat{\Delta}t_s$), which was previously defined in equation 4.7.

Then, the weighted gradient distance is summed up for each time step $t_s$ of the subsequence in the time serie $\mathcal{D}_i$ containing the similarity brush $\mathcal{GS}_k$, as shown in equation 4.8. The first time step is referred to as $t_L$ and the last one as $t_M$, as illustrated in figure 4.8. Note that an arbitrary number of time steps can be located between $t_L$ and $t_M$, and that the intervals $\hat{\Delta}t_L$ and $\hat{\Delta}t_N$, as well as the gradients of the brush and the time series, are truncated at the time steps $t_L$ and $t_M$, respectively.

[4] In their work on time series data mining Keogh and Pazzani [45] apply a similar approach using weighted partial differences for similarity measurement.

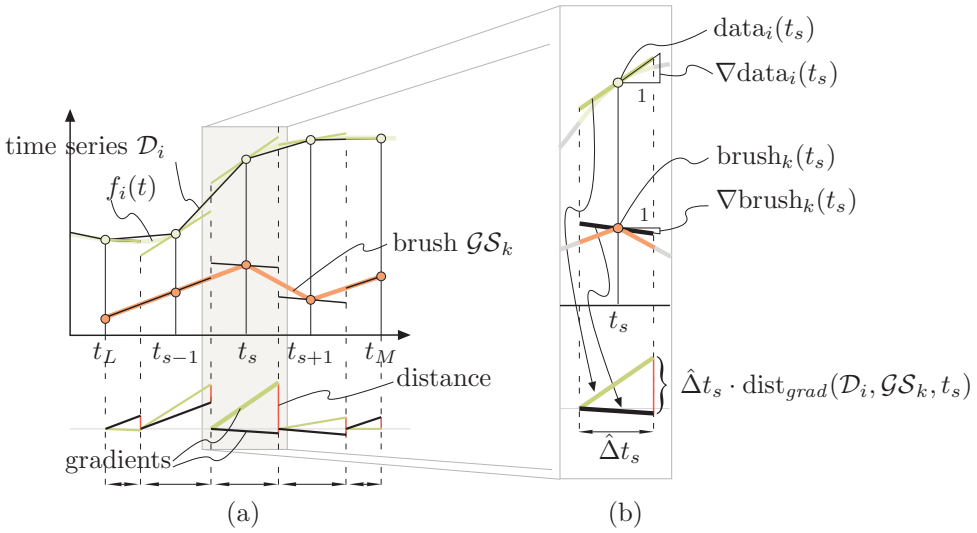



**Figure 4.8:** Evaluating a *gradient-distance-sum* (GDS) similarity brush: (a) The distances (red horizontal lines) between the gradients of the time series $\mathcal{D}_i$ (thick green lines) and of the similarity brush $\mathcal{GS}_k$ (thick black lines) are weighted properly, and added up. (b) Example distance calculation at a time step $t_s$. The gradients of the brush $\nabla\text{brush}_k(t_s)$ and of the time series $\nabla\text{data}_i(t_s)$ are calculated. Both are considered to be valid for the time interval $\hat{\Delta}t_s$, which is also the weight of the distance $\text{dist}_{grad}(\mathcal{D}_i, \mathcal{GS}_k, t_s)$. The latter is the absolute difference of both gradients.

**Generalization:** In the following the *general case*, where the sampling time steps of the brush $\tilde{t}_j$ (see Sec. 4.3.1) and of the time series $t_s$ do not correspond, will be considered and briefly discussed[5]: One can, for instance, calculate (sample) the gradients of the similarity brush at an arbitrary number of discrete sampling time steps $\tilde{t}_j$—including the control time steps $\hat{t}_l$, which are explicitly specified by the user (see Eq. 4.2). In other words, this means calculating all gradients $\nabla\text{brush}_k(\tilde{t}_j)$ from the sampled brush values, using central differences as formalized in equation 4.6. These brush gradients are again considered to be valid for a corresponding time interval $\hat{\Delta}\tilde{t}_j$. Then $\nabla\text{brush}_k(t_s)$ in (4.8) represents an *aggregated gradient* at a certain time step $t_s$ of the time series, which is calculated as a weighted sum of the brush gradients $\nabla\text{brush}_k(\tilde{t}_j)$ belonging to the associated interval $\hat{\Delta}t_s$. Thereby, the sum of the weights has to be equal to one. This can be formalized as

$$\nabla\text{brush}_k(t_s) = \frac{1}{\hat{\Delta}t_s} \cdot \sum_{\tilde{t}_j \in \hat{\Delta}t_s} \hat{\Delta}\tilde{t}_j \cdot \nabla\text{brush}_k(\tilde{t}_j),$$

where the intervals $\hat{\Delta}\tilde{t}_j$ are truncated at the borders of $\hat{\Delta}t_s$. Another approach would be to sample the GDS similarity brush $\mathcal{GS}_k$ at the time steps of the time series $\mathcal{D}_i$, however, this causes some precision problems.

[5] This can happen, for instance, when the brush (pattern) is automatically (and horizontally) shifted over the time series data in order to reveal cyclic patterns. This approach is subject of future research and is described briefly in Sec. 8.

As the number of time series $\mathcal{D}_i$, $i = 1, \ldots, N$ in the data set is commonly much greater than the number of similarity brushes $\mathcal{GS}_k$, $k = 1, \ldots, R$ (i. e., $N \gg R$), the evaluation of the GDS brush can be done very efficiently, only considering the time steps $t_s$ of the time series $\mathcal{D}_i$, and calculating the (aggregated) brush gradients $\nabla \text{brush}_k(t_s)$, e. g. in a preprocessing step. Thus, time steps of the time series which are wider spaced (and assumed constant), than the brush's time steps can be evaluated efficiently using the respective aggregated brush gradients.

**DOI Evaluation:** The two thresholds for the similarity-based DOI computation (see Eq. 4.5) can be specified by the user in the data domain of the time series. This allows an intuitive mapping of these thresholds to the visual representation of the similarity brushes. Thus, the gradient-distance-sum similarity brush represent a very intuitive and effective way of selecting time series according to their similarity to a user defined reference pattern (e. g., polyline). A more detailed discussion of the different similarity brushes is presented later in section 4.5.

### 4.4.3 Angular-Distance-Sum (ADS) Smooth Similarity Brushes

The *angular-distance-sum* (ADS) smooth similarity brush, illustrated in figure 4.9, is the second type of similarity brush presented in this thesis. ADS brushes are very similar to the gradient-distance-sum (GDS) similarity brush (see Sec. 4.4.2). However, instead of summing up the weighted distances of the gradients, as it was done in the evaluation of a GDS brush, now the *absolute differences* of the angles between an ADS similarity brush $\mathcal{AS}_k$ and a time series $\mathcal{D}_i$ are each weighted properly and then summed up. Finally, the sum is divided by the sum of the weights in order to calculate the average angular distance between the time series and the brush. The angles are calculated as the arc tangent of the respective gradients, where the fact has to be considered that the time and data values are usually given in different data domains, e. g., temperature is given in $°C$ and time is given in seconds. Therefore, the time, data and brush values are transformed to the same data domain, i. e., the gradients are normalized as described below.

The main difference between this and the gradient based distance measure is illustrated in figure 4.9 (a). Each one of two gradients (green and black thick lines in the left and right illustration) that have the same gradient distance (red lines) can enclose different angles (blue arcs). In the depiction left, where the two gradients have a steep slope, they comprise a relatively acute angle, whereas the two gradients that have a lower increase include a rather obtuse angle (right illustration). An observer would normally perceive the two gradients on the left to be more similar than the two on the right, depicted in figure 4.9 (a). This is the motivation for measuring similarity with the angles included between the time series and brush gradients in angular-distance-sum (ADS) similarity brushes.

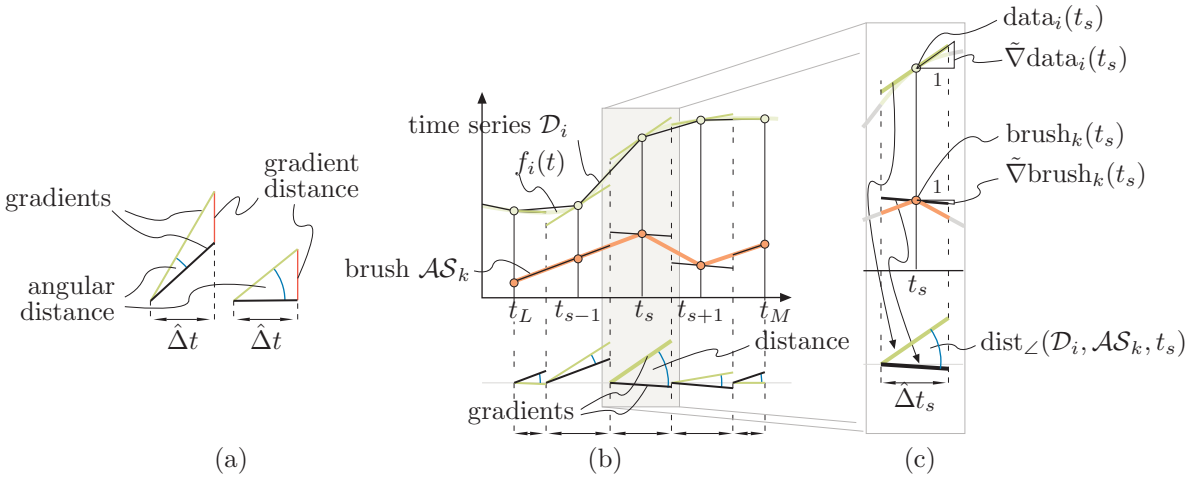



**Figure 4.9:** Evaluating a *angular-distance-sum* (ADS) similarity brush: (a) The angular distances (blue arcs) computed on the normalized gradients of the time series $\mathcal{D}_i$ (thick green lines) and of the the similarity brush $\mathcal{AS}_k$ (thick black lines) are weighted and added up. (b) Example angular distance calculation at a time step $t_s$. The normalized gradients of the brush $\tilde{\nabla}\text{brush}_k(t_s)$ and of the time series $\tilde{\nabla}\text{data}_i(t_s)$ are calculated. Both are considered to be valid for the time interval $\hat{\Delta}t_s$, which is also the weight of the angular distance $\text{dist}_\angle(\mathcal{D}_i, \mathcal{AS}_k, t_s)$. The latter denotes the angle between both gradients.

**Gradient normalization:** Before the arc tangent can be applied on the gradients, they need to be normalized. This is due to the fact that the time and data values are from different data domains, which was already mentioned. Accordingly, the gradients of the brush and the time series calculated using central differences (see Eq. 4.6) are scaled to the interval $[-1, 1]$ in order to restrict the resulting angle to the interval $[-45°, 45°]$:

$$\tilde{\nabla}\text{data}_i(t_s) = \frac{\hat{\Delta}t_{max}}{\hat{\Delta}x_{max}} \cdot \nabla\text{data}_i(t_s) \quad \text{and} \quad \tilde{\nabla}\text{brush}_k(t_s) = \frac{\hat{\Delta}t_{max}}{\hat{\Delta}x_{max}} \cdot \nabla\text{brush}_k(t_s),$$

where $\hat{\Delta}t_{max} = \max\{\hat{\Delta}t_j = \text{weight}(t_j) \mid t_j \in \mathcal{S}\}$, thus the maximum weight or time interval $\hat{\Delta}t_j$ of the time steps $t_j$ in the time dependent data set $\mathcal{S}$ (see Eq. 4.7), and $\hat{\Delta}x_{max}$ is the absolute distance between the maximum and minimum data value in $\mathcal{S}$, thus $|x_{max} - x_{min}|$ (see Eq.1.3). As a result, the maximal angle between the normalized gradients of a time series and a brush is 90 °.

**Evaluating Similarity:** With these normalized gradients, the angular distance between the subsequence of a time series $\mathcal{D}_i$ and an ADS similarity brush $\mathcal{AS}_k$, which is illustrated in figure 4.9 (b), can be formalized as $\text{dist}_\angle(\mathcal{D}_i, \mathcal{AS}_k) =$

$$\underbrace{\frac{1}{t_M - t_L}}_{\text{normalize}} \cdot \sum_{s=L}^{M} \hat{\Delta}t_s \cdot \underbrace{\left|\arctan\left(\tilde{\nabla}\text{data}_i(t_s)\right) - \arctan\left(\tilde{\nabla}\text{brush}_k(t_s)\right)\right|}_{\text{dist}_\angle(\mathcal{D}_i, \mathcal{AS}_k, t_s)}. \tag{4.9}$$

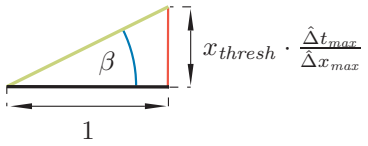



**Figure 4.10:** For the visual representation the angular threshold $\beta$ is mapped to the height of the similarity brush $x_{thresh}$ with respect to $\hat{\Delta}x_{max}$ and $\hat{\Delta}t_{max}$.

Thereby, at each time step $t_s$ of the time series $\mathcal{D}_i$ where the ADS similarity brush $\mathcal{AS}_k$ is contained, the weighted absolute difference of the angle included between the normalized gradient of the brush and the time series is summed up, namely $\tilde{\nabla}\text{brush}_k(t_s)$ and $\tilde{\nabla}\text{data}_i(t_s)$, respectively. This angular distance, which is emphasized in figure 4.9 (c), is denoted as $\text{dist}_{\angle}(\mathcal{D}_i, \mathcal{AS}_k, t_s)$. The two normalized gradients are both calculated for the time step $t_s$ and are considered to be representative for the associated time interval $\hat{\Delta}t_s$ in (4.9). As done for the GDS brushes each angular difference is weighted with the duration of the associated time interval ($\text{weight}(t_s) = \hat{\Delta}t_s$, defined in Eq. 4.7) before summing them up. In order to calculate the average angle between the time series and the brush, the sum is divided by the the sum of the respective weights $\hat{\Delta}t_s$, which is equivalent to the difference of the time values $t_M$ and $t_L$. The first time step is denoted as $t_L$ and the last one as $t_M$ (see Fig. 4.9 (b)). An arbitrary number of time steps can be located in-between $t_L$ and $t_M$.

As the angular-distance-sum is evaluated on normalized gradients in (4.9), it is independent of the visual representation of the time dependent data. When the user zooms into the visualization, the observed angles between the depicted time series and the brush change according to the respective scale factor. This does not affect the evaluation of the ADS similarity brush, however.

**DOI Evaluation:** The user-defined (angular) thresholds ($b_1$ and $b_2$) for the DOI evaluation are specified as the *average angle* between the normalized gradients calculated on the brush and data values and are applied in equation 4.5. For the visual representation of the angular thresholds the gradient normalization is reversed, i. e., $x_{thresh}(\beta) = \tan\beta \cdot \frac{\hat{\Delta}x_{max}}{\hat{\Delta}t_{max}}$. Thereby, the visual distance is computed in a right angle triangle where the adjacent has a length of 1 and the opposite is $x_{thresh} \cdot \frac{\hat{\Delta}t_{max}}{\hat{\Delta}x_{max}}$, both including a right angle (see Fig. 4.10).

### 4.4.4 Point-Sampled-Slope (PSS) Smooth Similarity Brushes

Both similarity brushes presented beforehand, namely gradient-distance-sum (GDS) and angular-distance-sum (ADS) brushes, require that the values of the time series are similar at all sampled time steps contained by the brush. In contrast to that, a *point-sampled-slope* (PSS) similarity brush evaluates the time series only at the control points explicitly specified by the user. These are tuples of values $(\hat{x}_l, \hat{t}_l)$, $l = 1, \ldots, P$, where $\hat{x}_l$ is the brush value defined in the data space of the time series, and $\hat{t}_l$ is the corresponding time value

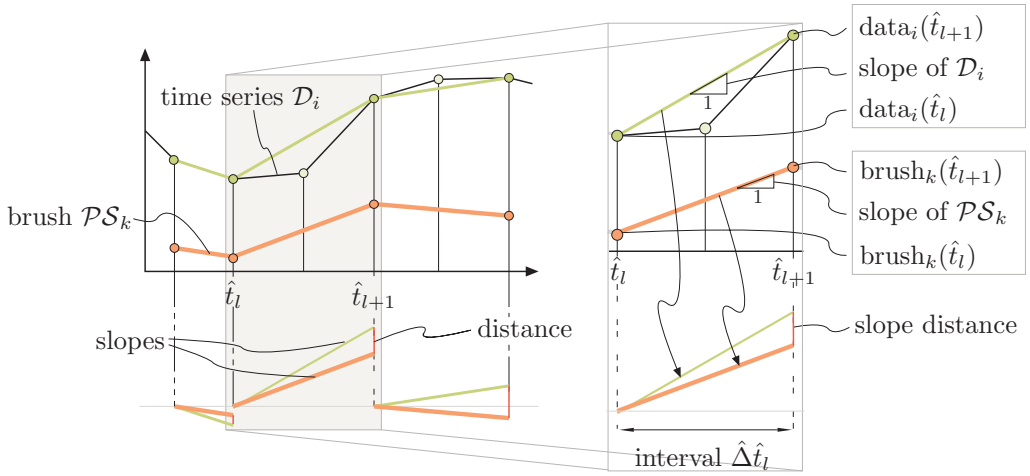



**Figure 4.11:** Evaluating a *point-sampled-slope* (PSS) similarity brush: (a) The slopes of the time series $\mathcal{D}_i$ (thick green lines) and of the similarity brush $\mathcal{PS}_k$ (orange) are only calculated at the control points explicitly specified by the user (orange dots). (b) Example slope distance calculation in-between the time step $\hat{t}_l$ and $\hat{t}_{l+1}$.

which is specified in the time domain (compare to Eq. 4.2). As long as the time series agree with the PSS brush in the control points, variations of the data values in-between are tolerated.

**Evaluating Similarity:** The distance between a PSS similarity brush $\mathcal{PS}_k$ and a time series $\mathcal{D}_i$ can be computed as illustrated in figure 4.11. In the process, the differences of the slopes of the time series and the brush, which are each computed between two *control time steps* of the brush $\hat{t}_l$ and $\hat{t}_{l+1}$, are summed up. As a result, the distance can be formalized as $\text{dist}_{points}(\mathcal{D}_i, \mathcal{PS}_k) =$

$$\sum_{l=1}^{P-1} \Big| \underbrace{\big(\text{data}_i(\hat{t}_{l+1}) - \text{data}_i(\hat{t}_l)\big)}_{\hat{\Delta}x_{i,l}} - \underbrace{\big(\text{brush}_k(\hat{t}_{l+1}) - \text{brush}_k(\hat{t}_l)\big)}_{\hat{\Delta}\hat{x}_{k,l}} \Big|, \tag{4.10}$$

where $\hat{\Delta}\hat{x}_{k,l}$ is the difference of the values $\hat{x}_{k,l} = \text{brush}_k(\hat{t}_l)$ and $\hat{x}_{k,l+1} = \text{brush}_k(\hat{t}_{l+1})$ of the brush $\mathcal{PS}_k$, and $\hat{\Delta}x_l$ is the difference of the associated data values $x_{i,l} = \text{data}_i(\hat{t}_l)$ and $x_{i,l+1} = \text{data}_i(\hat{t}_{l+1})$ of the time series $\mathcal{D}_i$. Thereby, the time steps $\hat{t}_{l+1}$ and $\hat{t}_l$ correspond to the user-defined control points.

**Generalization and DOI Evaluation:** The general case, where the time steps of the control points do not necessarily correspond to the time steps of the time series, will now again be considered. The time series data values in (4.10), namely $\text{data}_i(\hat{t}_{l+1})$ and $\text{data}_i(\hat{t}_l)$, are each computed using linear interpolation of the adjacent data values given in $\mathcal{D}_i$. The two thresholds used to derive a DOI from this metric (see Eq. 4.5) can be specified in the data domain of the attribute, displayed in the CurveView.

## 4.5 Discussion of the Brushing Approach

In addition to the visual data representation (see Chap. 3), the CurveView enables the user to interactively specify complex *time-dependent features* using brushing techniques. Features are specified in an interactive and iterative process and visualized immediately in all linked SimVis views using focus+context visualization. Based on the visual feedback, the user can further extend or refine the respective brushes. The combination of multiple brushes (within a view or in the hierarchical FDL tree) using fuzzy-logic operations (i. e., AND, OR, NOT) allows the specification of complex time-dependent features in SimVis.

The proposed *time steps brushes* are very useful, e. g., to restrict (or extend) specified features to a certain data range or to explore the evolution of time series that run through a certain area of the CurveView. Furthermore, similar trends and patterns within the time series data can be revealed using *gradient-based similarity brushes*. Each one of the different types of similarity brushes has its own characteristics which address different issues and tasks, as it is illustrated[6] in figure 4.12:

**Gradient-distance-sum brushes:** This type suites very well to select even time series that have a relatively low curvature. As it is shown in figure 4.12 (a) and (d) the highlighted time series are very similar to the shape of the user-defined pattern. The direct mapping of the threshold(s) to the height of the brush is quite intuitive. When the user, for instance, vertically moves the similarity brush above the selected curves, they are contained within the shape of the brush.

**Angular-distance-sum brushes:** This kind fits well to brush time series having a steep slope. Since the similarity is evaluated based on angles, the time series can be rather curved (see the red time series indicated by an arrow in figure (b)). The mapping of the threshold(s) is not as intuitive as with GDS brushes, since an averaged angular distance computed on normalized gradients is specified (see Sec. 4.4.3). However, the behavior when interactively altering the threshold(s) (e. g., by dragging) is very similar to the GDS brushes.

**Point-sampled-slope brushes:** This brush type is valuable for common tasks were the user is only interested in the slopes of the time series over certain periods of time. For instance, when brushing temperature curves that have a rising of $2 - 5°C$ over ten years, where it is irrelevant whether the time series vary in-between. Such an example is shown in figure 4.12 (c) and (f) where the time series indicated by the arrows vary from the bigger part of the highlighted curves. In this brush type, also the direct mapping of the threshold(s) to the height is relatively intuitive.

[6] For illustration purpose, the outer region of the brush (second threshold) is set to zero.

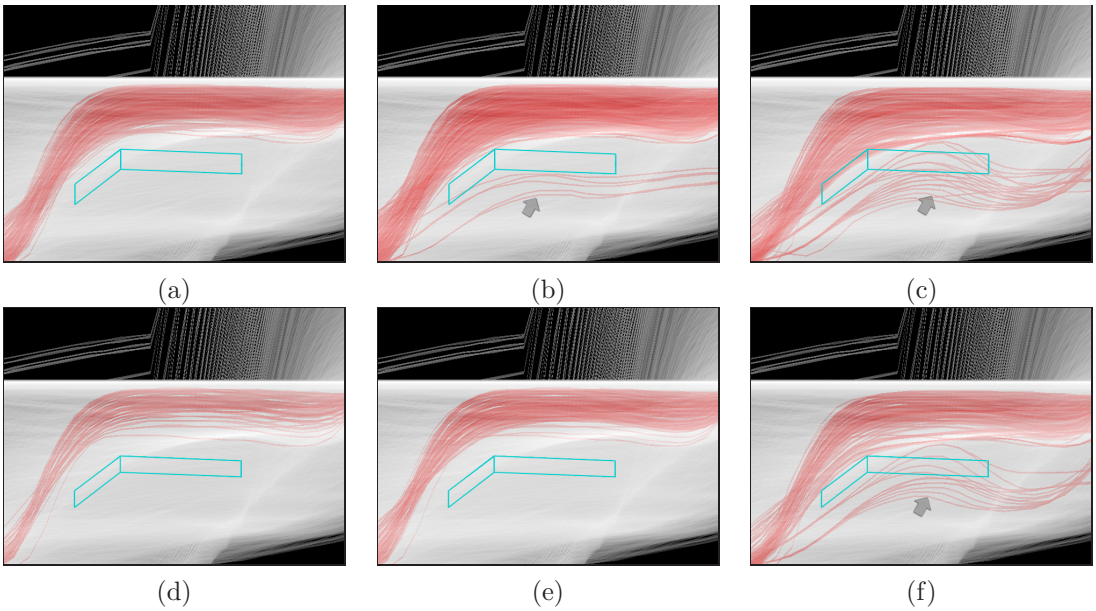



**Figure 4.12:** Comparing the types of gradient-based similarity brushes with different thresholds (i. e., in the top row a larger threshold is used). (a, d) gradient-distance-sum (GDS) brushes; (b, e) angular-distance-sum (ADS) brushes; (c, f) point-sampled-slope (PSS) brushes. Since the brushed time series (features) are visual outliers in areas of context information, the respective DOI values were amplified in order to make the features better discriminable.

# 5 Application Examples

In this chapter sample applications using the CurveView in combination with other linked SimVis views are described. In the context of *visual analytics* [83, 82] (see Sec. 2.1.1) and Keim's *visual analytics mantra* [40] (see Sec. 2.1.1), there are two issues, which are of major importance, when analyzing data visually and specifying features via interactive brushing. These are facilities for *interaction* (e. g., changing parameters affecting the visualization or navigating through the data) and *human factors* (e. g., intuition, expert and domain knowledge, the ability to detect visual structures and patterns in short time). Visual structures appear or disappear, for instance, when altering the visualization mapping of a CurveView (e. g., linear/logarithmic mapping, scale factor, constant offset, number of bins) or when features are specified via brushing and visualized using focus+context visualization. In this context, feature specification is an interactive and iterative process where the user searches for unknown and interesting patterns in the data, and furthermore aims to extract these features which will be described in the following.

In the following, a detailed visual analysis of the hurricane Katrina is presented (see Sec. 5.1). Then, an analysis of climate research data is briefly presented in section 5.2. Finally, the application aspects of the CurveView are discussed and summarized in section 5.3.

## 5.1 Interactive Visual Analysis of Hurricane Katrina

Hurricane Katrina was one of the strongest hurricanes ever recorded that made landfall in the United States. It hit the south coast near New Orleans, Louisiana, in August 2005, and caused enormous floods and fatal damage along the north-central Gulf Coast. More than 1,800 people lost their lives in the hurricane and the subsequent flooding[1]. The data set is provided by the Mesoscale and Microscale Meteorology (MMM) division at NCAR (National Center for Atmospheric Research).

In the simulation data set the position of the eye of the hurricane is centered over all time steps. This is done by centering the model at the the region with lowest pressure values. As a result the geographical context of the data set changes over simulation time, i. e., the land is shifted to keep the hurricane in the middle. The data set contains 73 time steps which are evenly spaced in one-hour intervals and 18 dimensions, such as wind speed, pressure, temperature or water vapor. Thereby, a static grid is used to calculate

[1] http://en.wikipedia.org/wiki/Hurricane_Katrina (accessed Sept. 2007)

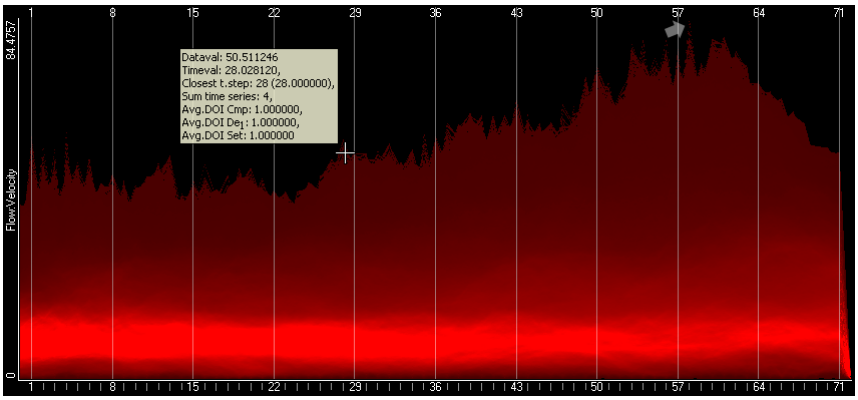



**Figure 5.1:** Visualization of the evolution of the *wind speed* (velocity) over time. A linear transfer function is applied, mapping density values (number of overlapping time series) to color intensities. Moreover, a constant offset is added, to emphasize the regions with high velocity (dark red) in the upper area of the View. The general data trends are depicted as a red band with high color intensity (lower area of the view). Applying an *info box* (yellow tooltip), further details about the original time series data can be inspected.

the simulation data at each time step. For the purpose of interactive analysis the original data set was downsampled to a static grid with 411,026 cells, i. e., a time series represents the evolution of the data in a spatial area (simulation cell) over time.

### 5.1.1 Visual Analysis of the Wind Speed using a CurveView

First of all, we want to analyze the evolution of the *wind speed* (velocity) over time, since a strong wind is an important and destructive characteristic of a hurricane. For this purpose, a CurveView is opened to depict the "Velocity" attribute of the hurricane data set, which is shown in figure 5.1. Thereby, a resolution of $256 \times 256$ bins per bin map is used, and a linear mapping of density to opacity values (see Sec. 3.3.2) is applied, where the scale factor is altered interactively by the user. Initially, the regions with high wind speeds (dark red) are barely visible in the CurveView. Therefore, a constant offset is added to the intensity values of the time series (using a slider), to emphasize the portion of the data, which is sparsely populated—the result is already depicted in figure 5.1. When using a CurveView, the observer gets a very good impression about the *general evolution* of the data values—and their distribution—over time. One is moreover enabled to make *quantitative statements* on the data, which is illustrated in the following.

#### Making Quantitative Statements on the Velocity Data

The *general trend* of the velocity data stays rather constant in the lower area of the CurveView in figure 5.1, while simulation time proceeds. This can be seen from the broad red band, depicted with high color intensity in the view, which is vertically centered around a velocity value of 8 meters per second (m/s). Furthermore, the time series with

*high wind speed*, which are located in the upper area of the view (above the trends) close to the respective maximum velocity values, are approximately evenly distributed. This (vertical) distribution of the velocity values at certain time steps can also be inspected, e. g., in a histogram view. However, using the CurveView gives a very good impression about the data evolution and (vertical) distribution over time. Considering, how the velocity data evolutes in figure 5.1, the maximum values vary from 23 to 59 m/s between the time steps (hours)[2] 0 and 24. Then the max. wind speed accelerates, and reaches its peak around hour 58 with 84.5 meters per second (indicated by the arrow). From hour 60 to 71, the high velocity decreases continuously, and then rapidly decelerates at the end of the data set.

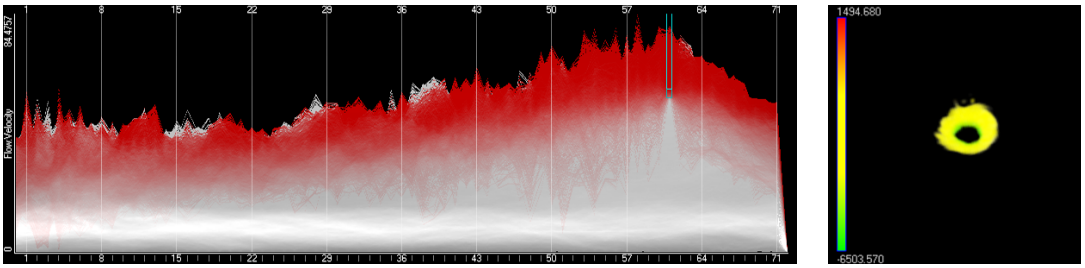
Enabling a tooltip (the yellow *info box*) in the CurveView in figure 5.1 one can explore further details of the original data located close to or at the current mouse position (represented as a white cross) in the visualization. These *details on demand* [79] in the info box are (see Fig. 5.1): the coordinates of the mouse pointer in data space ("dataval." and "timeval."); the closest time step to the cursor position and its time value; the number of time series passing through the respective vertical data interval (bin) at the closest time step ("sum time series")—this is calculated, adding up the bin counts in the respective row (or column) of the associated bin map; and the respective averaged DOI information of the feature nodes in the hierarchical FDL tree (see Sec. 2.1.2), aggregated for the time series running through the vertical interval (bin), i. e., the DOI values of the feature component, the feature description, and the feature set nodes.

To refine the information displayed in the info box or to increase the precision of the visual output the user can interactively alter the resolution of the bin maps. Furthermore, one can change the opacity mapping (e. g., linear/logarithmic mapping, alter scale factor, constant offset) in order to reveal the number of overlapping time series. In addition to that the user can navigate through the visualization by zooming and panning, enlarging interesting looking visual structures for more detailed observation.

#### Brushing Time-Dependent Features with High Velocity Values

In the following we want to focus on spatial grid cells (i. e., time series) where high wind speeds occur—this is an important feature of a hurricane or thunderstorm. Therefore, a smooth *time step brush* is created in the CurveView which depicts the velocity data (described above), the result is shown in figure 5.2 (a). Thereby, all time series with *high velocity* values at time step 61 are brushed and visually highlighted in red color. To put it another way, this means that all cells of the grid which have a data value belonging to the specified interval (at the certain time step), receive a high DOI value at all of their

[2] Only a few sample time steps are represented as long vertical lines crossing the CurveView in figure 5.1, while each time step is depicted as a tick below the time axis. Moreover, the number of the time steps is depicted above, and the respective time value below the view. Thereby, these values are equal in figure 5.1, i. e., the time-series data is regularly sampled in one-hour intervals.




(a) (b)

**Figure 5.2:** (a) High velicity values are brushed at time step 61; (b) the corresponding feature is visualized using a 3D view where the pressure values are color coded.

time steps[3]. The evolution of the time series, intersecting with the time step brush, can be examined in the view (see Fig. 5.2 (a)). Thereby, the DOI values are additionally emphasized, applying a gamma function with $\gamma \approx 0.4$ (see Eq. 3.8), otherwise the colors would be barely visible[4]. As one can see in figure 5.2 (a) the general trend of the selected time series (depicted in red color) is highly related to the maximum velocity values. This is because of the fact that the data set (i. e., the sampling grid) is aligned with the center of the hurricane. Furthermore, one can see that the brushed time series have a high variance, i. e., the velocity increases and decreases over simulation time (see the red outliers in Fig. 5.2 (a)). This is because of the circular winds around the center of the hurricane.

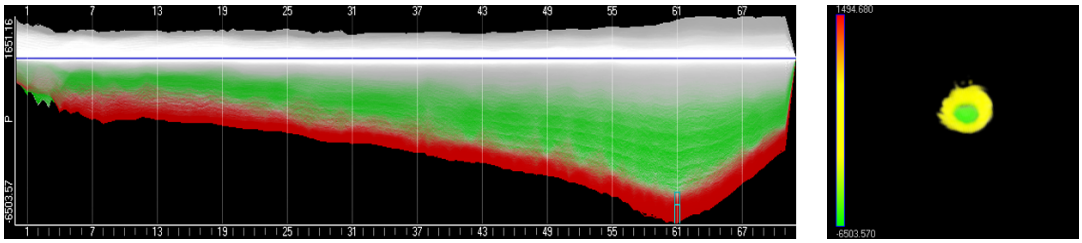
Using a *3D view*, one can see the spatial position of the selected time series at a certain point in the simulation time. The user can pass through the time steps of the simulation by the use of a slider and observe how the associated data in the 3D representation change. In the 3D view in figure 5.2 (b) the hurricane is shown at time step (hour) 61 where the respective pressure values are mapped to the color values of the data items. Thereby, green corresponds to low pressure, yellow to mean, and red to high pressure (see the color gradient, left). Moreover, the DOI information is mapped to the color saturation and opacity values of the data items, i. e., important data items (focus) are depicted with high opacity and saturated colors, while the rest (context) is displayed translucent and uncolored. As it is shown in figure 5.2 (b), the center of the hurricane (not visible or black) is surrounded by a ring of high velocity (selected in the CurveView) and relatively low pressure (yellow color).

### 5.1.2 Interactive Visual Analysis of the Center of the Hurricane

As an next step we visualize the center of the Hurricane Katrina, called the *eye*, and explore how it varies over time. Since a *low pressure* is an important feature of the center

[3] Remember that a time series represents the temporal evolution of a data attribute in a certain spatial area, namely the associated grid cell.

[4] Due to this non-linear amplification of the DOI information, the observer might get a wrong impression about the amount of relevance of the represented features. Therefore, it is recommendable to use a gamma value equal to one, when making quantitative statements on the features.




(a) (b)

**Figure 5.3:** (a) Low pressure values are brushed at time step 61; (b) the combined feature of high velocity and low pressure values is visualized using a 3D view where the pressure values are again color coded.

of a hurricane, the associated dimension is assigned to another CurveView (i. e., pressure), which is shown in figure 5.3 (a). Again, a linear opacity mapping and a constant offset are applied. In doing so, the general data trends are depicted with high intensity close to the zero line which is indicated as a horizontal blue line in the upper area of the view. Moreover, the low pressure values (already highlighted in color) in the bottom part of the view are emphasized, because of the constant intensity offset. Otherwise, these portions of the data would disappear in the visualization, as they are very sparsely populated by the time series (i. e., outliers). In addition to that, also the DOI information is strengthened, using a gamma value close to 0.5 (i. e., a square root function). Note, that the CurveViews shown in figure 5.2 (a) and 5.3 (a) are both child nodes of the same feature set node (in the FDL tree), which is important to mention in connection with the color coding of the specified features.

### Brushing Time Series with low Pressure Values

In figure 5.3 (a), time series with a *low pressure* value at time step 61 are selected by a time step brush, which is represented as rectangle, with (light green) inner and smooth outer region (dark green). The respective curves are emphasized in red color, furthermore, time series being brushed in the other linked CurveView (see Fig. 5.2 (a)), are highlighted in green color. Since *importance-driven color coding* is applied in the view (described in Sec. 2.1.2 and Sec. 3.3.2), the time series with *red color* (i. e., selected in the view itself being a child node of a feature description node) cover those with *green color* (i. e., selected in the feature set node by the other CurveView). Note, that the DOI information is additionally enhanced ($\gamma \approx 0.4$ in Eq. 3.8) to be better visible in the view. As it is shown in figure 5.3 (a), the trends of the red and green time series are highly correlated. Furthermore, the green curve form visual patterns among each other, which seem to be similar (in a subjective way). This issue will be further analyzed in section 5.1.3.

Again, we analyze the spatial relationship of the brushed features in a 3D view. Therefore, a fuzzy OR-combination of the features, which are specified in the two CurveViews, namely high velocity (see Fig. 5.2 (a)) and low pressure (see Fig. 5.3 (a)), respectively, is

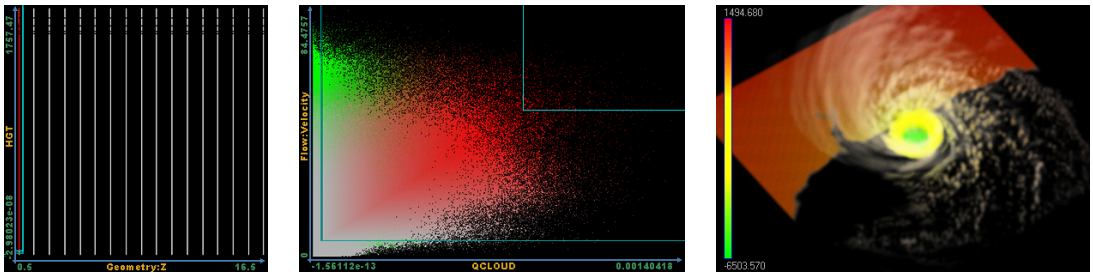
shown in the 3D representation of the data at time step 61 in figure 5.3 (b). Thereby, again the respective pressure values are encoded in color, and the DOI information is mapped to the color saturation and opacity of the 3D data items (as it is done in the previous 3D view in Fig. 5.2 (b)). The eye of the hurricane is shown as the green area in the middle of the view in figure 5.3 (b), which is surrounded by the area of high wind speed (yellow color). The later feature is specified in the first CurveView (see Fig. 5.2 (a)), and also depicted in the 3D view in figure 5.2 (b). The user can also enable and disable features in the FDL tree, and thereby observe the respective changes in the visualization.

#### Specifying other features for orientation

For orientation purpose, we want to depict the *geographical context* (i. e., the land), and the structures formed by *fast clouds* around the center of the hurricane, in the 3D data representation. The corresponding features are specified in attribute views being child nodes of the same feature set node as the two CurveView, i. e., the respective DOI values are combined using a fuzzy OR-operation. This approach, is similar to previous work done by Doleisch et al. [20], where hurricane Isabel is analyzed in SimVis, and will be described in what follows.

For the *land feature*, a 2D scatterplot [68] is opened (see Fig. 5.4 (a)). The attributes *height* ("Geometry Z"), representing the absolute height of the simulation cells, and the *relative height* of the surface ("HGT"), which is measured with reference to the sea level, are assigned to the horizontal and vertical axis of the scatterplot, respectively. Then, data items with a low absolute height, and (in addition to that), a relative surface height, which is larger than a certain value, are brushed in figure 5.4 (a), using a rectangular smooth brush. After specifying the geographical feature, now *fast clouds* are selected in another 2D scatterplot depicted in figure 5.4 (b). Here, an attributes representing the clouds ("QCLOUD"), and another representing velocity, is assigned to the two axes of the view. The data items are brushed using another rectangular smooth brush. Thereby, a large outer region (dark green color) is specified around the inner region of the brush (light green), where the DOI values change linearly form zero (context data located outside of the brush) to one (focus data located inside the inner region of the brush).

As a result, the visual structures representing the fast clouds around the center of hurricane Katrina are represented colorless (similar to white), which is shown in the 3D view in figure 5.4 (c). This corresponds to the fast clouds feature (see Fig. 5.4 (b)), where many data items receive a DOI value within $]0, 1]$—representing the smooth boundary between the focus and the context—, and are therefore depicted with low color saturation. Moreover, the land feature is shown in a dark orange color (high pressure) below the hurricane for orientation. Again pressure is color coded in figure 5.4 (c) (see the color bar, left) and the 3D view shows the hurricane at hour 61. This visual representation corresponds very much with pictures of hurricanes, well known from the media.




(a) (b) (c)

**Figure 5.4:** (a) The land is selected in a 2D scatterplot view; (b) fast clouds are selected using another scatterplot; (c) the combination of all specified features is depicted in a 3D view where the pressure values are color coded.

### 5.1.3 Revealing Interesting Time-Dependent Patterns and Trends

As it was mentioned in section 5.1.2, when analyzing the pressure data in the CurveView, several trends of colored time series are visible, which represent the specified features, and appeared to be similar (see Fig. 5.3 (a)). In the following we want to further explore these trends and visual structures, using the *similarity-based brushing* techniques presented in this thesis (see Sec. 4.4). As it was already mentioned at the beginning of this section, human intuition and the background knowledge of the expert using the application are of major importance, when specifying features and analyzing the data visually. In the context of visual analytics (see Sec. 2.1.1), feature specification is an interactive and iterative process, where the user searches for unknown and interesting patterns in the data, and furthermore extracts these features via brushing.

#### Brushing Similar Trends of Time Series in the Pressure Data

For the purpose of selecting *trends of time series*, which are similar amongst each other, a *smooth similarity brush* (see Sec. 4.4) is created in the CurveView, depicting the pressure data of the hurricane (see Fig. 5.5). The brush is represented in green color, and covers a certain (horizontal) period of time. The slope of the brush is altered interactively by the user, dragging the control points, which define the brush. In doing so, a time-dependent pattern is specified, and used for interactive feature specification. Thereby, time series being similar to the brush (i. e., they have a similar slope or gradient), are selected smoothly. The associated feature is visually reflected immediately, highlighting the respective time series in color (using focus+context visualization). Moreover, the user can interactively modify the inner (light green) and smooth outer region (dark green) of the similarity brush, by dragging the respective edges at the control points of the brush. In doing so, the thresholds used for the evaluation of the similarity, are altered (see Sec. 4.3.3). Again, these modifications affect the feature specified by the smooth similarity brush, which is depicted instantly, using focus+context visualization (also in the linked views).

In the CurveView shown in figure 5.5 the user has zoomed into the pressure data, using

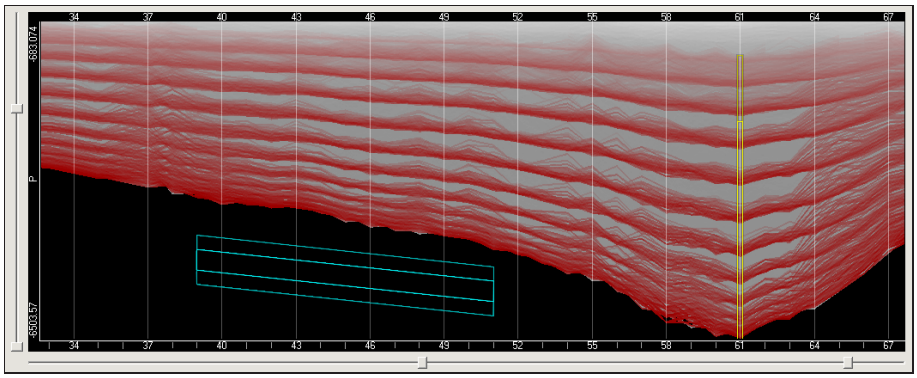


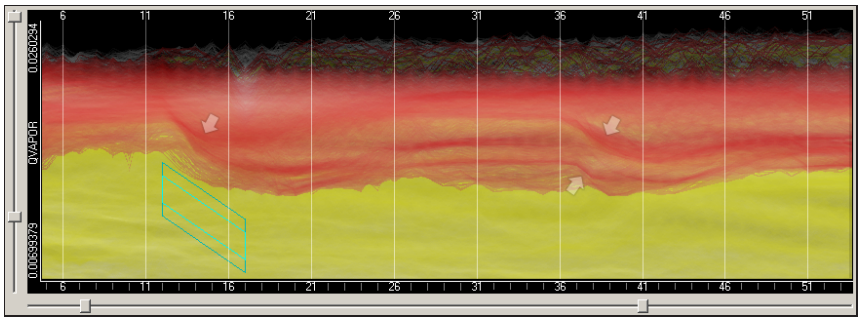

**Figure 5.5:** Similar trends in time-dependent pressure values are selected using a similarity brush.

the horizontal and vertical sliders. Applying a *gradient-distance-sum* (GDS) similarity brush (see Sec. 4.4.2), a pattern is specified for similarity-based brushing, which has a similar slope as the outer hull formed by the minimum pressure values in the data set. Moreover, the resulting feature is refined to those time series, which have a low pressure value—this is a feature of the eye of the hurricane—at time step 61, by the use of a smooth *time step AND-brush* (yellow rectangle). Several traces of similar pressure curves are visible in the view—this phenomenon results from the simulation process and is explained below—, which are highlighted in red color. As one can see from the figure, the traces escalate at time step 63/64, which indicates that the hurricane makes *landfall*. Refining the specified feature to one of these traces (e. g., using another time step AND-brush), one can see in the linked 3D view, that the associated data belongs to a certain height level of the center of the hurricane. This is due to the fact, that the time-dependent pressure data is calculated for different height layers during the simulation process. Thereby, the time series within one layer are continuously distributed, however, the time series of different height levels are unevenly distributed (see the white space between the similar time series). Using the similarity brush, the typical trends of the pressure values within the center of the hurricane are emphasized. Without the brushing facilities, these patterns would be hard to discover.

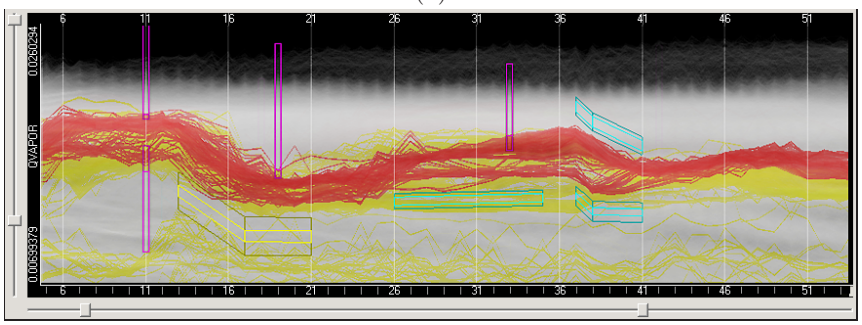
### Further Similarity-Based Brushing Results

When analyzing other dimensions in the hurricane data set using a CurveView, one comes across other interesting *visual structures* formed by the time series. This means, that one *localizes* possible (unknown) *features* such as data trends, patterns, or outliers. As an example, the *water vapor* is analyzed, which builds clouds or fog in a condensed form. As it is done in the previous section, a feature is specified in a scatterplot view, brushing the simulation cells, which are close to the *land*, by the use of a rectangular smooth brush (see Fig. 5.4 (a)). As a result, a visual trend is visible in the CurveView depicting the water vapor ("QVAPOR").

Using a *point-sampled-slope* (PSS) smooth similarity brush (see Sec. 4.4.2), one aims




(a)




(b)

**Figure 5.6:** Brushing trends in the evolution of the water vapor. (a) a reoccurring pattern (indicated by the arrows) is discovered using a similarity brush; (b) the feature is refined using further similarity and time step brushes.

to select the above mentioned visible trend, as it is done in figure 5.6 (a). This type of similarity-based brush tolerates variations of the time series in-between its control points. As a result, the time series, following a downward trend during a certain time span in the water vapor data, are brushed smoothly. The resulting feature is highlighted in color using focus+context visualization. The time series shown in red color (i. e., specified in the feature component node) are brushed in both view (i. e., the CurveView and the 2D scatterplot), those in yellow color (i. e., the feature description node) are only brushed in the CurveView. As it is shown in figure 5.6 (a), the specified downward trend reoccurs about 24 hours (time steps) later, which is indicated by the arrows. This is due to the fact, that the amount of water vapor decreases during night over land, which is also visible when inspecting the clouds in a 3D view.

Applying further brushes in the CurveView, which are combined in the FDL tree using fuzzy logic operations (see Sec. 4.2.2), one can interactively refine the selection to the desired trends and thereby specify a complex feature (see Fig. 5.6 (b)). The similarity brush previously created is converted to an AND-brush. The two visible trends around time step 38 are selected interactively using further similarity brushes (green), which are combined in the FDL tree by a fuzzy OR-operation. Using time step brushes several data ranges are excluded from the feature (i. e., NOT-brushes depicted pink). The trends visible in figure 5.6 (a) are specified as a complex time-dependent feature in (b).

## 5.2 Visual Analysis of Climate Research Data

In corporation with the Wegener Center for Climate and Global Change in Graz, Austria, the SimVis system—including the CurveView—was successfully applied to the analysis of climate research data. In this context, a case study done by Ladstätter et al. [52] has been presented as a poster at the *3rd Intern. Workshop on Occultations for Probing Atmosphere and Climate (OPAC-3)*. Thereby, scatterplots and CurveViews are used to brush derived time-dependent attributes (e. g., linear trends or signal-noise-ratio (SNR)). The goal was to localize robust indicators for monitoring climate changes as well as deficiencies within climate research data set. The topic and the results are briefly discussed in the following (compare to Ladstätter et al. [52]).

### 5.2.1 Indicators of Climate Change

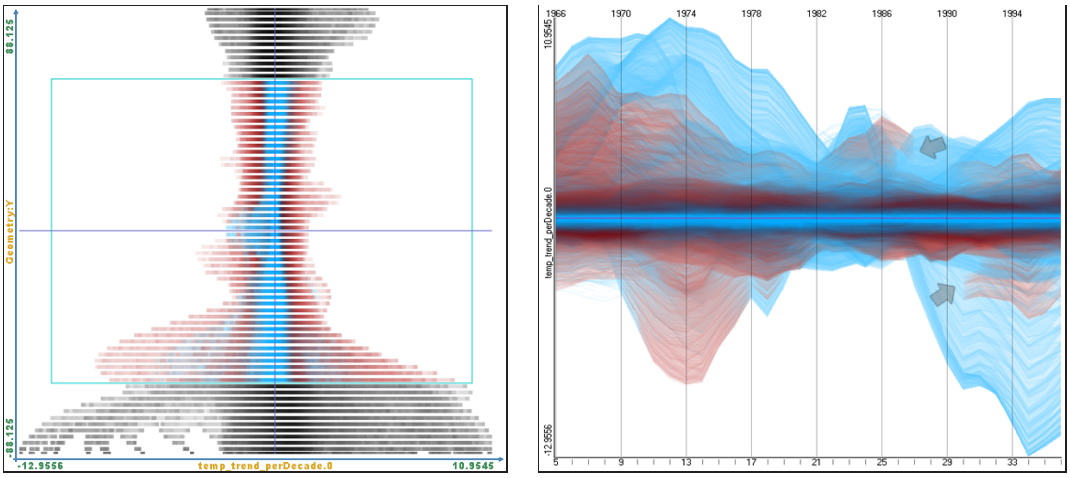
In *climate research*, one is commonly interested in the temporal evolution of certain attributes (e. g., temperature, specific humidity and refractivity) in the *upper troposphere and lower stratosphere* (UTLS) domain (5–35 km), which is considered to be a sensitive region with respect to climate change. Since the second half of the last century, *measurements* are performed and due to technical improvements the resulting data becomes increasingly accurate (e. g., the Radio Occultation (RO) technology available since 1995 using Global Navigation Satellite System (GNSS) signals [51]). In this context, *climate models* data sets (e. g., ECHAM5) aim to provide long-term scenarios, and *reanalysis* data sets (e. g., ERA-40) combine the measured data from various sources (e. g., satellite data, radiance data).

In order to find promising indicators for the monitoring of climate change (in the UTLS region), two data sets, namely ECHAM5 and ERA-40, are explored interactively using the SimVis system. The basic parameters (e. g., temperature, refractivity, specific humidity and geopotential height) provide rather trivial and well-known interrelations. Thus, derived parameters are considered for analysis purpose, such as *linear trends* calculated as moving 10-years-difference (similar to central differences) or the *signal-noise-ration* (SNR), which is defined as the ratio of the linear trend to the detrended (i. e., the trend is subtracted from the original data) standard deviation [52]. Interesting features with respect to robust indicators consist of high values for the linear trend and a relatively good SNR.

### 5.2.2 Analyzing the ERA-40 Reanalysis Data Set

In the following, selected examples from the analysis of the ERA-40 reanalysis data set are briefly presented and discussed (see the work done by Ladstätter et al. [52]). Since earlier measurements were not as accurate as nowadays (e. g., radio occultation), the data set comprises know deficits in southern high latitudes.

In figure 5.7 an example exploration of summer seasons in the time period 1966 to 1997 is shown, where the CurveView is used in a passive manner. High absolute signal-to-noise (SNR) values of temperature are selected over the whole time span using a

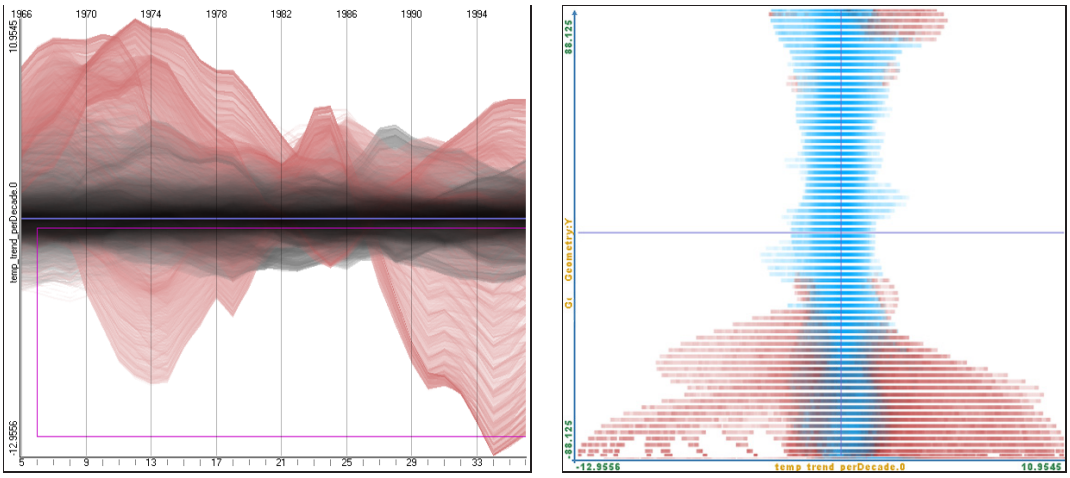



(a) (b)

**Figure 5.7:** Analysis of the evolution of temperature in climate research data: (a) The distribution of the derived linear temperature trends (x-axis) vs. geographical latitude (y-axis). High SNR values are brushed in another scatterplot, and are highlighted in red in (a). Here, the latitude region between 60°N and 60°S is brushed, since the ERA-40 data set shows known deficits in southern high latitudes. (b) The evolution of the derived trend of the temperature from 1966 to 1997 is shown in the CurveView, where a logarithmic opacity mapping is applied in order to emphasize outliers. The latitude and the SNR feature are depicted. Most of the outliers (blue) do not belong to this feature. Images taken from Ladstätter et al. [52].

rectangularNOT-brush in a 2D scatterplot. The resulting feature is highlighted in red in another 2D scatterplot showing the distribution of the derived trend per decade (explained above) of the temperature (x-axis) vs. latitude (y-axis) as shown in figure 5.7 (a). A certain range of latitude is brushed (60°N to 60°S) where the data set is supposed to contain robust indicators for climate change. In the CurveView in figure (b) the temporal evolution of the derived temperature trend is shown. Furthermore, the latitude and the SNR feature are depicted using color coding. Most of the outliers (blue) in the upper and lower part of the view are not highlighted, i. e., they do not correspond to the brushed range of latitude or their absolute SNR is too small. There is also some discontinuity of the highlighted (red) curves (indicated by the arrows) which results from the SNR feature.

In the CurveView in figure 5.8 the derived temperature trends with high variations are selected using a horizontal similarity NOT-brush (gradient-distance-sum, depicted pink). Accordingly, the regions with outliers are highlighted in red, while the general data trends (vertically centered around the zero line) are depicted in black. The corresponding feature is visualized using the 2D scatterplot with the same attributes as in figure 5.7 (a). In figure 5.8 (b), the bigger part of the highlighted data items (red) appears at high southern latitudes (bottom, left and right), where the data set comprises known deficits.




(a) (b)

**Figure 5.8:** (a) The derived temperature trends are shown in the CurveView. Time series with high variations are selected using a similarity NOT-brush. (b) The distribution of the derived linear temperature trends (x-axis) vs. geographical latitude (y-axis) are visualized using a scatterplot. The bigger part of the highlighted data items (red) appear occurs at high southern latitudes.

## 5.3 Discussion

The CurveView allows the user to quickly get insight into the *evolution* of the data over time. Different types of multi-variate (or multidimensional) data can be assigned to the view. The *specification* of the visualization (i. e., some parameters) can be altered in order to allow the user to focus on different aspects of the time-dependent data set. One can, for instance, emphasize *general data trends* using a linear opacity transfer function, which mapps the number of overlapping time series (density) to opacity values (i. e., color intensity). On the other hand, low populated regions in the visualization can be accented (i. e., visual outliers) applying a logarithmic opacity transfer function or by adding a constant offset to the intensity values of the time series. Using focus+context visualization both aspects (trends and outliers) of the data can be represented within one view. This gives a good overview of the data evolution over time.

In addition to the aspects related to the visualization, the CurveView enables the user to interactively specify complex *time-dependent features* applying interactive brushing techniques. By the use of *similarity-based brushes*, trends of time series being similar to a user-defined pattern are brushed smoothly. Thereby, the pattern can be sketched directly in the CurveView. Time series that pass through a specified interval at a certain time step, can be selected using *data range-driven brushing* (i. e., time step brushes). Thereby, brushing is an interactive and iterative process where the specified features are visualized immediately in all linked SimVis views using focus+context visualization. As the brushes are combined within a view using *fuzzy-logic operations* (i. e., AND, OR, NOT), and/or

hierarchically combined within the feature-definition-language (FDL) tree, very complex time-dependent features can be specified and visualized in SimVis and in the CurveView.

# 6 Implementation Details

## 6.1 The SimVis Framework

SimVis [15, 21, etc.] consists of a core application and an *extensible framework*, which allows the development of so-called *plugins* providing new functionalities to the system (e. g., additional views, data converters). The software is designed to run on multiple platforms (e. g., Linux, Windows) and is written in the *C++* programming language. Furthermore, *OpenGL*[1] and NVIDIA's *C for graphics* (Cg) [24] are commonly applied for data visualization, where the graphics hardware is used in order to display the data at interactive frame rates. The extensions to SimVis are compiled to separate *dynamic link libraries* (DLLs), and are loaded at program start-up.

As it was already mentioned in section 2.1.1, the features specified and visualized in multiple bidirectionally *linked views* are organized in a hierarchical *feature definition language* (FDL) tree (managed by the SimVis framework), which can also be saved and loaded. At start-up, a view has to register at the FDL tree and provide certain basic functionalities to be seamlessly integrated into SimVis, i. e., some graphical user interface (GUI), methods to display the specified features using *focus+context visualization*, or functions that respond to *degree-of-interest* (DOI) updates, to name a few. Moreover, attribute views have to provide *smooth brushing* techniques for interactive feature specification. Each of these attribute views is represented as a feature component node affecting the FDL tree.

As it was pointed out before, *user interaction* as well as the *real-time visualization* of the data, are both key issues in visual analytics (see Sec. 2.1.1). Hence, several (crucial) tasks in SimVis are decoupled using *multiple threads*. For instance, the GUI events and the graphical data representation in a view are commonly executed in separate threads, in order to respond immediately to the actions of the user (e. g., dragging a slider, zooming into the data, panning). Moreover, DOI updates in the FDL tree (e. g., when a brush is altered) can take up to a few seconds, and are therefore executed in other threads. Another important features of the SimVis system, is a low-level *memory manager*, which performs caching operations and swaps out blocks of data from main memory to the hard disk when the system runs out of memory. This is due to the fact, that the amount of data handled during an analysis session can be very high (e. g., several gigabytes).

[1] Open Graphics Library (OpenGL), see http://www.opengl.org/

## 6.2 The CurveView Plugin

The CurveView is written as a plugin (DLL) for the SimVis system, and is available as a feature component node in the FDL tree. Therefore, it provides brushing functionalities affecting the tree (see Chap. 4), and enhanced techniques to interactively visualize time series data (see Chap. 3). Moreover, the CurveView makes use of the basic functionalities provided by the framework, such as the loading and saving of the corresponding nodes in the FDL tree, or the loading of large time-dependent data sets. As it is done in other views, the depiction of the data, the GUI events, and the evaluation of the brushes are executed in separate threads, in order to decouple these crucial tasks. Moreover, graphics hardware is used allowing the real-time visualization of the time-dependent data.

### 6.2.1 Implementation of the Visualization Approach

Several techniques are applied, in order to archive enhanced interaction and interactive frame rates during rendering. When, for instance, parts of the FDL tree are modified (i. e., the DOI values) or the user interactively alters some visualization parameters while the rendering is being performed (in another thread), it has to be finished as fast as possible. Therefore, different tasks when depicting the data are performed separately, and each of them can be skipped, proceeding with the next step. These tasks are: the binning of the time series; the rendering of the binned data to the DOI density map; and the final depiction of this map.

**Binning:** When aggregating the time series data, the *2D (DOI) bin maps* are computed in software, where 32 bit are used per bin count and 8 bit per averaged DOI value (see Sec. 3.2). Thereby, the amount of data can also become very large, e. g., about 1 GB at a resolution of $1024 \times 1024$ bins per (DOI) bin map and 100 time steps. Thefore, the SimVis memory manager is applied for data allocations, which performs low-level caching or swaps out data blocks, when the system runs out of memory.

**Creating the DOI Density Map:** For the *real-time visualization* of the binned data, the respective geometric primitives (polygons) are depicted using the features of modern graphics hardware (available since NVIDIA GeForce 6 or ATI Radeon 9000). First, the data is rendered to a high-precision floating point texture (16 or 32 bit), which represents the *DOI density map* (see Sec. 3.3.1), using *vertex arrays*[2]. By the application of another feature called *floating point framebuffer blending*, the different color channels of the texture can be used to aggregate the binned information, i. e., the Red-Green-Blue (RGB) channels sum up the respective DOI information, and the Alpha channel aggregates the number of overlapping time series (density).

**Depicting the DOI Density Map:** In a final step, the DOI density map (i. e., the texture)

[2] Vertex arrays are a feature of modern graphics hardware where the geometric data is stored in an array, which can be processed efficiently.

is rendered to the output device, in order to depict the time series data. Using *programmable shading* (compare to Fernando and Killgard [24]), a Cg fragment program is applied, which performs certain arithmetic operations or texture look-ups directly on the graphics hardware, for each output pixel in the CurveView. Thereby, the transfer functions described in section 3.3.2 are evaluated, which map the respective density value (from the texture) to the opacity value (color intensity), or perform the DOI enhancement and the importance-driven color coding.

### 6.2.2 Allowing enhanced User Interaction

When for instance the DOI information is updated, the number of bins is altered or a different dimension is assigned to the CurveView, the time-dependent data and/or the associated DOI values have to be re-binned (see step 1). Thereby, only the *visible (DOI) bin maps* are processed in order to save time. To keep the respective information persistent, a flag classifies whether the map is up-to-date or not. On the other hand, when changing the view specifications (i. e., zooming, panning, switching between linear or logarithmic mapping, altering the intensity offset or scale factor), commonly only the rendering has to performed and the binned information—if it is up-to-date—can be *re-used* (see step 2).

The *rendering* of the binned information to the high-precision floating point texture (DOI density map) can take a while, even though only a few hundred milliseconds (see Sec. 6.3). However, for the purpose of visual analytics this can be to long (compare to Novotny and Hauser [67]), thus, two textures are used. The current data is always rendered to one of these textures, if the task cannot be completed (e. g., interrupted by user interaction) the previously created texture is used (and eventually scaled or translated) in the final rendering step (step 3). In doing so, the user gets *continuous visual feedback* allowing the enhanced navigating or altering of a brush without losing the orientation. Furthermore, certain areas in the (DOI) bin maps can be skipped when the user has zoomed into the visualization. Since the respective geometry it is not visible, it need not be sent to the graphics hardware (see the gray squares in Fig. 3.10 in Sec. 3.4.1), which saves rendering time.

### 6.2.3 Hardware Optimized DOI Evaluation

The evaluation of the similarity-based brushes in the CurveView can be highly optimized, when the distance is computed using special component-based vector arithmetics. *Streaming SIMD Extensions* (SSE) is a SIMD (Single Instruction, Multiple Data) instruction set, which is available on commonly used processor architectures, e. g., on Intel processors[3] upwards the Pentium III and on AMD processors[4] (3DNow). Thereby, the same instruction (e. g., adding, subtracting, multiplying, square root, min/max operations) can be performed simultaneously on four 32-bit floating point values in special registers added to the

[3] http://www.intel.com/
[4] http://www.amd.com/

CPU architecture. Using this instruction set, the distance calculation could be advanced.

## 6.3 Performance Evaluation

The performance of the CurveView was evaluated, measuring the respective time required to complete a certain task (e. g., rendering, binning, DOI evaluation). Two large time-dependent data data sets were used for testing: the hurricane Katrina data set, which contains 411, 026 time series given on 73 time steps; and a large climate data set provided by the Wegener Center with 921.600 time series and 67 time steps. Only the cases were considered, where the respective task could be finished without being interrupted (e. g., by updates caused by other threads handling user interaction or DOI updates) in order to measure the required time to perform the complete task. In the cases were certain tasks could not be finished, however, the DOI density map (texture) previously created was visualized (taking less than a millisecond), i. e., a visual feedback was continuously provided to the user.

The results of the evaluation of the visualization approach are shown in table 6.1. As one can see from the measurements, the rendering time highly increases when the resolution ($L \times L$) of the (DOI) bin maps is doubled, i. e., $\mathcal{O}(L^2)$. The resolution, however, does hardly affect the binning time, which is linearly dependent on the number of time series. Furthermore, rendering and binning are both linearly dependent on the number of visible time steps (i. e., the number of (DOI) bin maps to be depicted). In table 6.2 the time required to evaluate the similarity-based brushes using SSE is shown. Thereby, the angular-distance-sum (ADS) brush takes much longer to be evaluation than the gradient-distance-sum (GDS). Moreover, the required time linearly depends on the number of time steps contained in the brush. Comparing brush evaluation and binning time, one can see that the most crucial (time demanding) task is the binning of the data. However, when navigating through the data representation or changing the visualization parameters, the binned data can be re-used. Moreover, the number of bins can be reduced in order to enhance interaction.

| *Data set* | *Task* | *#vis.time steps* | *required time (sec)* | |
|---|---|---|---|---|
| | | *#bins/map* → | 256 × 256 | 128 × 128 |
| Hurricane Katrina (411,026 time series) | Rendering | 65 t.steps | 0.098 | 0.029 |
| | | 35 t.steps | 0.045 | 0.015 |
| | Binning | 65 t.steps | 1.739 | 1.485 |
| | | 35 t.steps | 0.882 | 0.770 |
| Climate Data Set (921,600 time series) | Rendering | 65 t.steps | 0.220 | 0.064 |
| | | 35 t.steps | 0.127 | 0.038 |
| | Binning | 65 t.steps | 3.159 | 2.965 |
| | | 35 t.steps | 1.751 | 1.594 |

**Table 6.1:** Performance Evaluation.

| *Data set* | *brush type* | *required time (sec)* | |
|---|---|---|---|
| | *#time steps* → | 25 t.steps | 50 t.steps |
| Hurricane Katrina (411,026 time series) | gradient-distance-sum | 0.045 | 0.093 |
| | angular-distance-sum | 0.380 | 0.189 |
| Climate Data Set (921,600 time series) | gradient-distance-sum | 0.105 | 0.201 |
| | angular-distance-sum | 0.429 | 0.849 |

**Table 6.2:** Performance when evaluating similarity-based brushes.

# 7 Summary

## 7.1 Introduction

Massive and complex amounts of time-dependent information arise in various areas of business, science and engineering such as climate research, medicine or finance. The data sets commonly result from measurements, modeling or the simulation of dynamic processes and contain multiple attributes (i. e., dimensions) evolving over time. In this context, analysts are commonly interested how the data changes over time in order to predict future developments. Thus, they want to investigate general *data trends*, to discover interesting structures and patterns as well as data items which differ from the main trends, called *outliers*. In addition to that, one is often interested in the relationships between different attributes (dimensions) of the data. In order to gain insight and knowledge from such large and complex data sets, the science of *visual analytics* [83, 82] (aka. interactive visual analysis) combines sophisticated methods from different research disciplines. Using interactive, visual and analytical techniques, analysts are enabled to create, verify or reject hypotheses based on the data.

However, when dealing with data sets that contain a large number (e. g., several hundred thousands or millions) of time series, the application of standard visualization approaches such as bar- or line charts (compare to Harris [29]) is commonly not sufficient and results in overcrowded and visually cluttered displays. Here, the important information (e. g., interesting patterns or features) is simply hidden by the massive amount of data. Moreover, the depiction of this data can take up to a few seconds—even on modern graphics hardware—which is also not satisfactory for the purpose of interactive visual analysis [1].

The objective of this thesis is to present the *CurveView* for the enhanced visual analysis of multidimensional time series data. The approach is seamlessly integrated into *SimVis* [15, 21, etc.], a multiple-views system usually dealing with complex time-dependent simulation results. *Smooth brushing techniques* [17] allow the user to select certain interesting subsets of the data (*features*) in an intuitive manner, where fuzzy classification [96] is applied. Time series are brushed, based on their *similarity* to a user-defined pattern (e. g., polylines) directly outlined in the view. Moreover, the user can select time series running through a certain area of the CurveView. The data is depicted using *focus+context visualization* techniques [30], where important or selected portions of the data (focus) are visually accented, while the rest of the data (context) is shown in a less prominent style. Using customizable transfer function, visual structures, patterns and outliers can be emphasized in the visualization. Moreover, the amount of data is reduced—while preserving

its characteristics—using 2D binning techniques [67], which allow the depiction of the data at interactive frame rates. Using multiple linked SimVis views complex relationships and features can be visually analyzed.

## 7.2 Related Work

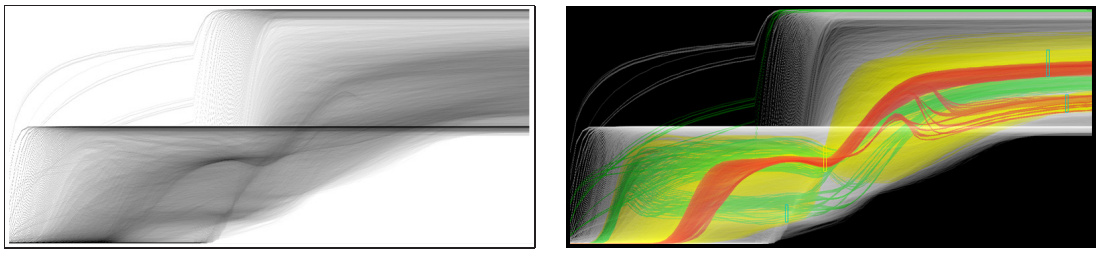
A large variety of publications deal with the visualization and analysis of *time-oriented information* (see the work of Aigner et al. [2, 1] for an overview). In order to reduce visual cluttering, Johansson et al. [37] use high-precision *density maps* [89] in their work on parallel coordinates that count the number of primitives (*density*) that pass though each screen pixel. User-defined *transfer functions* [55] are applied that map density to opacity values in final output. This allows to emphasize and visually discriminate different portions of the data (e. g., major trends, outliers). Novotny and Hauser [67] apply 2D *bin maps* for data aggregation (reduction) in parallel coordinates, in conjunction with outlier detection and clustering within the maps. Depicting the binned information allows the focus+context visualization [30] (described in the introduction) of large data sets showing data trends while preserving outliers at interactive frame rates.

Several applications exist that allow the analysis of time-dependent data using interactive brushing or querying techniques. Feature visualization and specification via linking and brushing in multiple views is an integral part of *SimVis* [15, 21, etc.] and has also been used for the visual analysis of multidimensional time-dependent simulation data [19]. Thereby, scalar degree-of-interest (DOI) values from the unit interval represent the importance of the data items. The values are altered using *smooth brushing* [17] where a linear border region is assumed between focus (DOI = 1) and context (DOI = 0), and focus+context visualization is applied to represent features.

Konyha et al. [49] introduce *Line Brushes* to select function graphs (e. g., polylines representing time series), which intersect with a simple line segment drawn in the view. The *TimeSearcher* [36, 35, 9, and others] is an application especially designed for the visual analysis of time series using TimeBoxes or *Angular Query Widgets* (compare to Hochheiser and Shneiderman [36, 35]). The later, select time series having a similar slope on a sequence of time steps (compare to *angular brushing* [32]). A further enhancement of the TimeSearcher [9] allows the *similarity-based querying* for temporal patterns. *QuerySketch* [88] enable the user to outline the shape of a pattern used for querying directly in the view. Inspired by this technique, *QueryLines* [77] allows the graphical specification of approximate queries, where soft constraints and preferences are used for fuzzy pattern description and ranking of the query results, respectively.

## 7.3 Focus+Context Visualization of Large Time Series Data

The proposed approach allows the interactive *focus+context visualization* of large time series data sets. An intuitive visual metaphor is used for the representation of the time se-



(a) (b)

**Figure 7.1:** Focus+context visualization of a data set containing 30.000 time series. (a) The number of overlapping time series is encoded in the color intensity using a logarithmic mapping; (b) brushed features are encoded in color.

ries, which is comparable to simple *line graphs* [29]. The visualization process consists of three steps: Initially, the massive amount of data is reduced using *2D binning* techniques adapted from Novotny and Hauser [67]. Thus, the depiction of the binned information instead of the raw data can be done in real time; Using graphics hardware the data is rendered to a high-precision *density map* (i. e., textures); Finally, customizable *transfer functions* are applied on the texture in order to reveal visual structures such as outliers or general trends (compare to Johansson et al. [37]). Moreover, the DOI information representing the importance of the data items is incorporated into the visualization using *color coding*. Thus, the user is enabled to visually assess how many time series run through a certain region on the screen and how important this information is (see Fig. 7.1).

### 7.3.1 Data Binning

Using *2D bin maps* [67], the interval $[x_{min}, x_{max}]$ containing all the time-dependent values of a data set is subdivided into a user-defined number of non-overlapping bins of equal width $w$ (see Fig.7.2). The data values given at two successive time steps $t_s$ and $t_{s+1}$ are then mapped to the Euclidean coordinates of the bins in an associated 2D bin map, which counts the number of time series aggregated in a bin (*bin count*) and, therefore, represents a discrete 2D frequency-based representation (compare to a histogram) of the data. The associated scalar DOI information **doi**, which is given at different hierarchies of a *feature definition language* (FDL) tree in SimVis, is also incorporated into the binning process using separate *DOI bin maps*, which sum up the respective importance of the data values given at $t_s$ and $t_{s+1}$, respectively. This process is illustrated and further described in figure 7.2. Aggregating the whole data set, means to transform the data given at each of two successive time steps into a reduced representation using associated (DOI) bin maps.

### 7.3.2 Rendering the Binned Data to a high-precision DOI density map

Using the features of modern graphics hardware, the binned data can be rendered efficiently to a high-precision *DOI density map* (RGBA texture). Each element in the texture

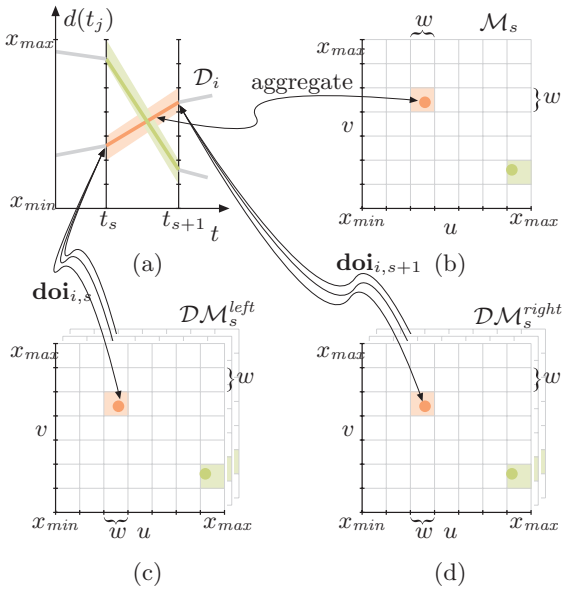
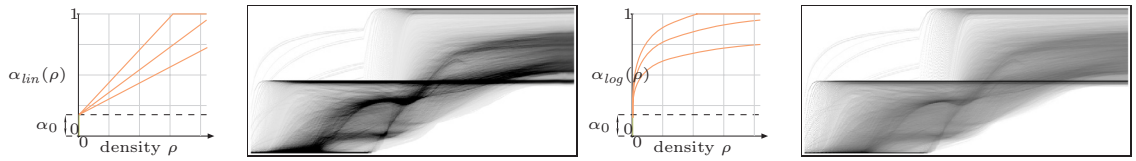



**Figure 7.2:** Aggregating the number of line segments connecting the time-dependent values and the associated DOI information in a bin map and two DOI bin maps, respectively: (a) At each time step, the data interval $[x_{min}, x_{max}]$ containing all values of the data set is subdivided into a user-defined number of bins of equal width $w$. The line segments connecting the values given at two successive time steps $t_s$ and $t_{s+1}$ of a time series $\mathcal{D}_i$ in (a) are aggregated in an associated discrete 2D *bin map* $\mathcal{M}_s(u, v)$ in (b). Each of the *DOI bin maps* (c, d) consists of several layers for the DOI vectors **doi**, which contain DOI values from different hierarchies in SimVis' feature definition language (FDL) tree. The associated tuples of DOI values $\mathbf{doi}_{i,s}$ given at time step $t_s$ are averaged in a bin of the *DOI bin map* $\mathcal{DM}_s^{left}$ (c), and the tuple of DOI values $\mathbf{doi}_{i,s+1}$ given at the subsequent time step $t_{s+1}$ are averaged in a bin in the $\mathcal{DM}_s^{right}$ (d).

corresponds to a pixel on the display screen and consisting of a *density value* $\rho_{u,v}$ (alpha channel of the texture) counting the number of primitives running through it, and a vector $\mathbf{d}_{u,v}$ (color channels), which accumulates the *DOI values* given at different hierarchies in the FDL tree. Thus, for each bin in each (DOI) bin map an associated parallelogram is rendered to the texture. Thereby, the coordinates of the graphical primitive are defined by the time steps and the vertical start and end position of the bin (i. e., coordinates). The bin count is assigned to the opacity value of the rhomboid affecting the density values, and the DOI values are encoded in the color values summed up in $\mathbf{d}_{u,v}$.

### 7.3.3 Depicting the DOI density map using Focus+Context Visualization

When depicting the DOI density map, the density information $\rho$ is represented as the intensity (opacity $\alpha$) of the respective primitives using a customizable linear or logarithmic transfer function (compare to Johansson et al. [37] and see Fig.7.3). This allows the observer to easily count or estimate the number of time series—represented as polygons—




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

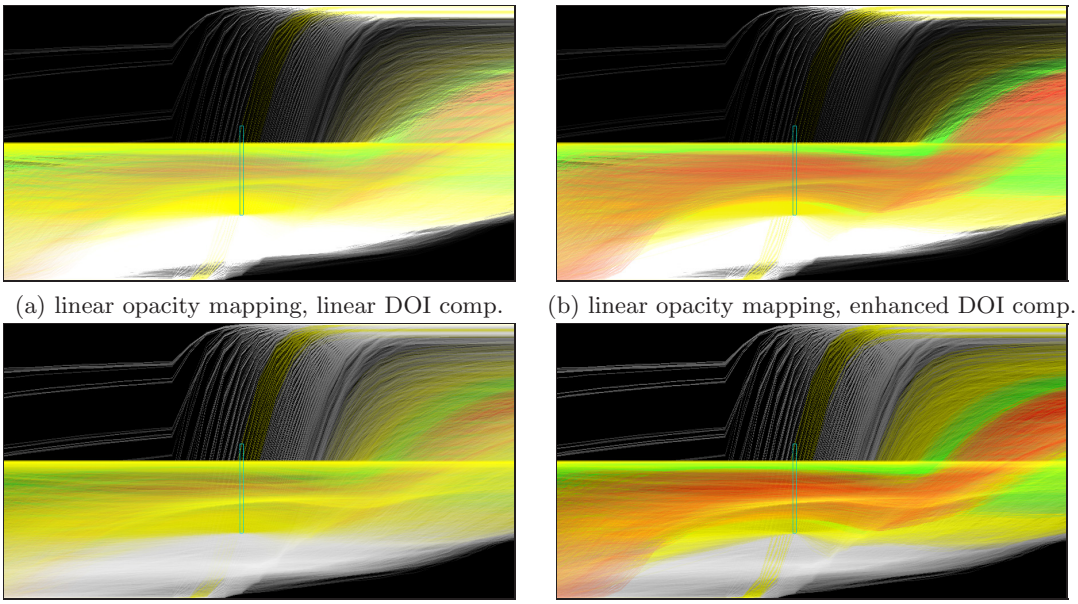
**Figure 7.3:** Mapping density $\rho$ to opacity values: (a) linear transfer function $\alpha_{lin}(\rho)$ with different slopes (i. e., scale factors) and constant offset $\alpha_0$; (b) trends are highlighted according to the scale factor, and outliers are preserved, because of the offset; (c) logarithmic transfer function $\alpha_{log}(\rho)$ using different scale factors (orange curves) and offset $\alpha_0$; (d) the outliers are highlighted, while preserving the trends.

that pass through a certain region in the visualization. In this context, it is important that the user can interactively change the opacity mapping to reveal the number of overlapping items (compare to Fekete and Plaisant [23]). In addition to that, the DOI values aggregated in the DOI density map are visualized using importance-driven *color coding* (compare to Doleisch et al. [15, 17, 21, etc.]). Thereby, more important colors obliterate those less important, according to the respective hierarchy of the DOI values in the FDL tree. DOI values can be further amplified in order to emphasized the *DOI information* from the DOI density map, i. e.,

$$\mathbf{doi}_{u,v} = \left(\frac{\mathbf{d}_{u,v}}{\rho_{u,v}}\right)^{\gamma}, \quad \text{where} \quad \mathbf{doi}_{u,v} \in [0,1]^3 \quad \text{and} \quad \gamma \in ]0,1]. \tag{7.1}$$

In figure 7.4 different mappings are illustrated. Thereby, a *linear* opacity transfer function is applied in figure (a) and (b), which maps the number of time series, running through a pixel, to the respective opacity (intensity) value. Thereby, a steep slope is used in order to emphasize the main portion of the time series, and to make the color coding (feature visualization) visible. However, this has the drawback, that visual structures (due to intensity) in dense areas of the visualization are hardly discriminable. When scaling down the linear mapping the trends are revealed, while the outliers disappear. On the other hand, a *logarithmic* opacity mapping is used in figure 7.4 (c) and (d). Here, the outliers are enhanced, while preserving the general data trends in the visualization. However, due to the non-linear mapping, one can not intuitively estimate the number of overlapping time series from the color intensities. For this purpose, a linear mapping is recommendable.

In figure 7.4 (a) and (c) a *linear* function is used (i. e., $\gamma = 1$ in Eq. 7.1), to calculate the normalized DOI values, applied in the importance-driven color coding. Accordingly, the red and green time series (i. e., the features) are hardly discriminable in the visualization. On the other hand, a *square-root* function is applied (i. e., $\gamma = 0.5$ in Eq. 7.1) in the CurveViews shown in figure (b) and (d). Therefore, the DOI values (and the features) are represented in an enhanced manner, i. e., the red and green time series are revealed. However, this does not correspond to the real amount of important time series in the visualization, which is better represented using a linear mapping.



(a) linear opacity mapping, linear DOI comp.
(b) linear opacity mapping, enhanced DOI comp.
(c) log. opacity mapping, linear DOI comp.
(d) log. opacity mapping, enhanced DOI comp.

**Figure 7.4:** Comparison of different opacity transfer functions and DOI enhancement: In the upper row a linear mapping of density to opacity values is applied, while a logaritmic mapping is applied in the bottom row. In the left column the normalized DOI information is representation, while this information is shown in an enhanced manner in the right column.

## 7.4 Brushing Time-Dependent Features

In addition to the visual data representation, the CurveView enables the user to interactively specify complex *time-dependent features* using brushing techniques. This is done in an interactive and iterative process where the features are visualized immediately in all linked SimVis views using focus+context visualization. Based on the visual feedback, the user can further extend or refine the respective brushes. The combination of multiple brushes (within a view or in the hierarchical FDL tree) using fuzzy-logic operations (i. e., AND, OR, NOT) allows the specification of complex time-dependent features in SimVis. The CurveView provides *time step brushes*, which can be used to select time series running through a certain region of the view, i. e., a specified data interval at a time step (compare to *line brushes* [49]). Moreover, time series can be brushed according to their similarity to a user-defined pattern, which is described in the following.

### Similarity-Based Brushing of Derived Attributes

The idea of the proposed *similarity-based brushing* approach is that the user sketches a time-dependent pattern directly in the CurveView by specifying an arbitrary number of control points. The points are then interconnected by a polyline which represents the shape of the *similarity brush* used for querying (see Fig 7.5). Time series are classified smoothly according to their similarity to the pattern, i. e., an adequate degree-of-interest (DOI)

value is assigned to the respective data items.

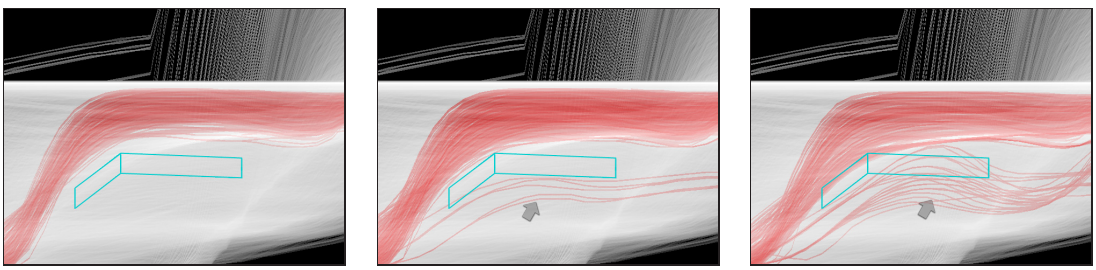
The approach uses the sum of *absolute differences* of the *gradients* or the *angles* between the time series data values and the brush in order to quantifying similarity (i. e., distance calculation) and to deal with the problem, when an constant offset is added to all data values of the brush or time series. In the context of smooth brushing, the resulting distance is then compared to two thresholds discriminating focus from context, where gradually increasing/decreasing DOI values are assumed at the boundary between the focus and context. The sum of absolute differences forms a useful compromise between a relatively low error rate and a distance measurement incorporating all values of the pattern and the subsequence of the time series. Moreover, the linear behavior of the distance metric fits very well with the boundary between the focus and context portion of the data in smooth brushing. As the time series are not necessarily sampled at evenly spaced time steps, the respective differences have to be weighted properly during evaluation[1].

Similar trends and patterns within the time series data can be revealed using *gradient-based similarity brushes*. Three different types of similarity brushes are described in this work, which each have its own characteristics and address different issues and tasks, as it is illustrated[2] in figure 4.12:

**Gradient-distance-sum (GDS) brushes:** This type evaluates similarity based on the absolute differences between the gradients of the brush and the time series, which are each weighted (to compensate for unevenly-spaced time series) and summed up. Thus, the highlighted curves (red) in figure 7.5 (a) are very similar to the shape of the user-defined pattern. This type is well suited for brushing even time series that have a relatively low curvature. The threshold(s) used for similarity evaluation can be direct mapped to the height of the brush, which is quite intuitive. When the user,

[1] In their work on time series data mining Keogh and Pazzani [45] apply a similar approach using weighted partial differences for similarity measurement.

[2] For illustration purpose, the outer region of the brush (second threshold) is set to zero.



(a) (b) (c)

**Figure 7.5:** Comparing the types of gradient-based similarity brushes. (a) gradient-distance-sum (GDS) brush; (a) angular-distance-sum (ADS) brush; (c) point-sampled-slope (PSS) brush. Since the brushed time series (features) are visual outliers in areas of context information, the respective DOI values were amplified in order to make the features better discriminable.

for instance, vertically moves the similarity brush above the selected curves, they are contained within the shape of the brush.

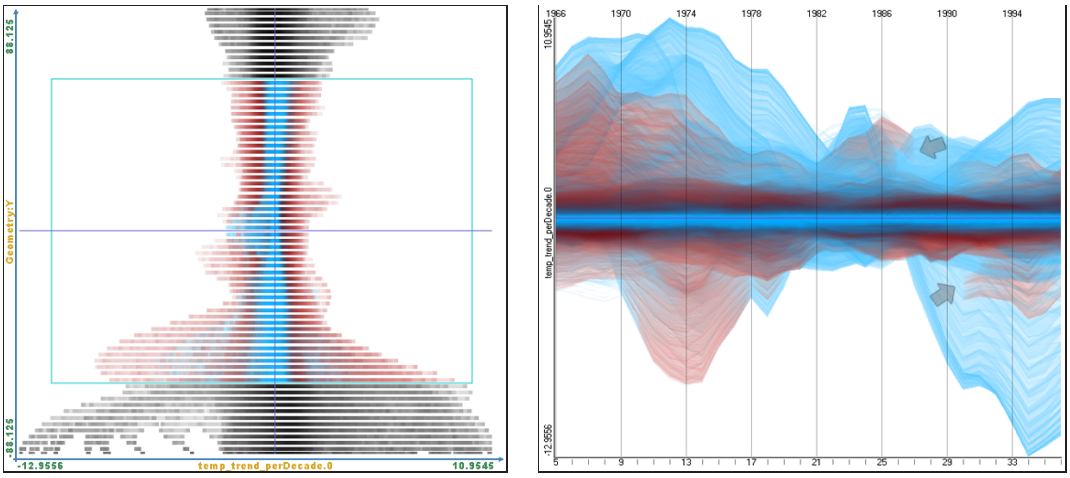
**Angular-distance-sum brushes:** This kind of brushes uses the respective angles between the time series and the brush (i. e., the arc tangent of the respective gradients) to evaluate similarity. Thereby, the fact has to be considered that the time and data values are usually given in different data domains. The brushed time series (red) can be rather curved, which is indicated by an arrow in figure (b). This type also fits well for querying time series that have a steep slope. The user specifies an averaged angular distance which is used for similarity evaluation.

**Point-sampled-slope brushes:** This type evaluates similarity only at the user-defined control points of the brush, where the respective slopes are computed. Hence, the time series in-between the control points can vary (indicated by the arrow in Fig.7.5 (c)). This type is valuable for common tasks where the user is only interested in the slopes of the time series over certain periods of time, e.g., when brushing temperature curves that have a rising of 1-2% over ten years. The threshold(s) can also be mapped directly to the height of the brush, which is relatively intuitive.

## 7.5 Application Examples

In corporation with the Wegener Center for Climate and Global Change in Graz, Austria, the SimVis system—including the CurveView—was successfully applied to the analysis of climate research data. The goal of a related case study done by Ladstätter et al. [52] was to find robust indicators for climate change as well as deficiencies within climate research data set. Derived parameters of certain attributes (e. g., temperature, refractivity or specific humidity) are considered for analysis purpose, such as *linear trends* calculated as moving 10-years-difference or the *signal-noise-ration* (SNR), which is defined as the ratio of the linear trend to the detrended (i. e., the trend is subtracted from the original data) standard deviation. CurveViews and 2D scatterplots are used to brush the derived attributes in order to localize robust indicators for climate change. In this context, interesting features consist of high values for the linear trend and a relatively good SNR.

In figure 7.6 an example exploration of a reanalysis data set (ERA-40) where measurements from different sources, such as satellite or radiance data are combined. Only summer seasons in the time period 1966 to 1997 are considered, where the CurveView is used in a passive manner. High absolute signal-to-noise (SNR) values of temperature are selected over the whole time span using a rectangular NOT-brush in a 2D scatterplot. The resulting feature is highlighted in red in another 2D scatterplot showing the distribution of the derived trend per decade (explained above) of the temperature (x-axis) vs. latitude (y-axis) as shown in figure 5.7 (a). A certain range of latitude is brushed (60°N to 60°S) where the data set is supposed to contain robust indicators for climate change. In the CurveView in figure (b) the temporal evolution of the derived temperature trend is shown.




(a) (b)

**Figure 7.6:** Analysis of the evolution of temperature in climate research data: (a) The distribution of the derived linear temperature trends (x-axis) vs. geographical latitude (y-axis). High SNR values are brushed in another scatterplot, and are highlighted in red in (a). Here, the latitude region between 60°N and 60°S is brushed, since the ERA-40 data set shows known deficits in southern high latitudes. (b) The evolution of the derived trend of the temperature from 1966 to 1997 is shown in the CurveView, where a logarithmic opacity mapping is applied in order to emphasize outliers. The latitude and the SNR feature are depicted. Most of the outliers (blue) do not belong to this feature. Images taken from Ladstätter et al. [52].

Furthermore, the latitude and the SNR feature are depicted using color coding. Most of the outliers (blue) in the upper and lower part of the view are not highlighted, i. e., they do not correspond to the brushed range of latitude or their absolute SNR is too small. There is also some discontinuity of the highlighted (red) curves (indicated by the arrows) which results from the SNR feature.

## 7.6 Implementation

The CurveView is integrating into the existing framework of the SimVis system [15, 21, etc.] and uses several of its features (e. g., memory management, bidirectional linking between views, crucial tasks are performed in separate threads). It provides additional brushing functionalities to SimVis affecting the FDL tree, and performs different tasks in separate threads (e. g., hanlding GUI events, evaluating the brushes).

The generation of the bin maps is realized in software. For the *real-time visualization* of the binned data, the respective geometric primitives (polygons) are depicted using the features of modern graphics hardware (available since NVIDIA GeForce 6 or ATI Radeon 9000) such as *floating point framebuffer blending* in order to create the DOI density map. The rendering of the map is realized using programmable shading

(e. g., evaluating transfer functions in a fragment program). Moreover, two high-precision textures are used, the current data is rendered to one of them, if the task is interrupted by another thread (e. g., DOI update) the previously stored texture is depicted. In doing so, the user gets *continuous visual feedback* allowing the enhanced navigating or altering of a brush without losing the orientation.

The evaluation of the similarity-based brushes in the CurveView could be highly optimized, computing the distance using *Streaming SIMD Extensions* (SSE)[3] available on commonly used processor architectures (e. g., Intel, AMD). Thereby, the same instruction (e. g., adding, subtracting, multiplying, square root, min/max operations) can be performed simultaneously on four 32-bit floating point values in special registers added to the CPU architecture.

## 7.7 Conclusions

The interactive visual analysis of time-dependent information is a challenging task, when very large data sets containing multiple dimensions and several hundred thousands or even millions of data items have to be visualized and analyzed at interactive frame rates. The *binning approach* (adapted from Novotny and Hauser [67]) has prooven to be very useful for the purpose of data reduction, allowing the real-time visualization of the data.

Using *focus+context visualization* (using color and intensity) the observer gets a very good impression about the evolution of the data over time and about the features interactively specified by the user. By the application of customizable transfer function a high degree of flexibility is provided to the user. Thus, *general data trends* can be revealed in the visualization as well as regions containing only a few time series, called *outliers*.

*Interactive brushing* techniques allow the specification of complex time-dependent features where multiple brushes are combined using fuzzy logic operations (AND, OR, NOT). Patterns of trends can be outlined directly in the CurveView. Time series being similar to this patterns are then brushed smoothly using fuzzy classification. Moreover, time step brushes can be applied to select time series that run through a certain area in the CurveView.

[3] Single Instruction, Multiple Data (SIMD)

# 8 Conclusions and Future Work

The interactive visual analysis of time-dependent information is a challenging task. This is especially true, when very large data sets containing multiple dimensions and several hundred thousands or even millions of data items have to be visualized and analyzed at interactive frame rates. The *binning approach* (adapted from Novotny and Hauser [67]), which was integrated into the visualization, has proven to be very useful for the purpose of data reduction. It allows the real-time visualization of a large number of time series (e. g., $500.000 - 1.000.000$) given at about 75 time steps. The approach is easily scalable to different tasks and data sets. One can, for instance, advance the level of details in the visualization (e. g., for presentation purpose) by incrementing the resolution of the (DOI) bin maps. On the other hand, the user can reduce the resolution in order to shorten the time required for rendering and binning, while interacting with the application.

By the application of *focus+context visualization* (using color and intensity) the observer gets a very good impression about the evolution of the data over time and about the features interactively specified by the user. Using a customizable transfer function the number of overlapping time series with respect to the overall number (i. e., density) is mapped to the brightness of the respective pixel. Thus, *general data trends* can be revealed in the visualization as well as regions containing only a few time series, called *outliers*. Moreover, the features specified in multiple views are color coded according to their relative importance. The features representation can also be amplified using another (non-)linear transfer function.

Interactive *brushing techniques* allow the user to specify complex time-dependent features where multiple brushes are combined using fuzzy logic operations (AND, OR, NOT). In this context, *patterns of trends* can be outlined directly in the CurveView. Time series being similar to this patterns are then brushed smoothly using fuzzy classification. Moreover, *time step brushes* can be applied to select time series that run through a certain area in the CurveView. The interrelations between the specified features in multiple time-dependent dimensions can be analyzed visually using *multiple linked views* in SimVis, which show different aspects (i. e., dimensions) of the data.

### Discussion

When interacting with the CurveView a relatively high amount of *adjustable parameters* is provided to the user (e. g., the thresholds of the similarity brushes, a linear or logarithmic opacity mapping, the customizable transfer functions). On the one hand, this makes the approach very flexible and applicable for different tasks and data sets, i. e., the user

can adjust certain parameters according to his/her requirements. On the other hand, an unexperienced user can get confused by the high number of parameters in the beginning. Therefore, useful default settings are applied initially, which provide relatively good results in common cases, e. g., the specifications of the visualization when opening a new CurveView or when assigning a new attribute; or default thresholds when creating a new similarity brush.

There are also more and less *important parameters* affecting the CurveView approach. Relatively important parameters are, for instance, the inner region of the brush (used for similarity evaluation); the transfer function's scale factor and the constant offset; or the feature enhancement (color coding). On the other hand, the smooth outer region (second threshold) of the similarity or time step brushes or the number of bins used for aggregating the data set are less important and are rather used for fine tuning.

In most cases, *straightforward strategies* exist in order to cope with certain problems in the visualization. For instance, when the trends in the time series visualization are not visible in densely populated areas of the display it is recommendable to alter the scale factor of the opacity transfer function. Furthermore, when the outliers are not represented in the visualization one can change the constant offset or apply a logarithmic opacity mapping. If the color coded features are barely visible, one can amplify the respective DOI values using a non-linear transfer function. In case where the system's response time to user interaction is to slow, it is also beneficial to reduce the number of bins per map. On the other hand, the visual precision (i. e., granularity) can be enhanced increasing the number of bins (e. g., for presentation purpose).

However, it is generally challenging to (quantitatively) *evaluate and compare* different approaches and applications for interactive visual analysis. This is due to the fact, that the search for certain features or the adjustment of the view's specification (affecting the visualization) is commonly an interactive and complex process, where human intuition and domain knowledge are very important. User studies can be performed, for instance, where certain tasks have to be fulfilled by the test persons. However, since the proposed CurveView approach is used by our collaboration partner, namely the *Wegener Center for Climate and Global Change* in Graz, we have received very positive feedback from the experts using the application, which was also incorporated into the development process. In this context, the approach was successfully applied in a case study on climate research data (see Ladstätter et al. [52]), which was presented at the 3$^{rd}$ Intern. Workshop on Occultations for Probing Atmosphere and Climate (OPAC-3).

### Future Work

The current *similarity-based brushing* approach has the drawback that the brush is only evaluated at the time steps where it is originally defined. However, in some cases it can make sense to search for a reference pattern over all time steps of the time series, for instance to discover *cyclic behavior*. For this purpose, the brush can be automatically

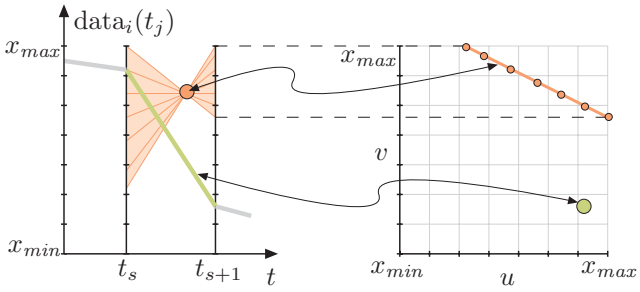



**Figure 8.1:** Duality between the screen space and the 2D bin maps: (a) The green time series is transformed to a point in the bin maps (b). On the other hand, all lines running through the orange point (pixel) in (a) are transformed to points lying on a straight line in the bin map.

shifted over the time series data, starting with the first and ending with the last time step. Thereby, similarity is evaluated at each translation step. The resulting DOI values are then assigned to the involved time steps where the respective DOI information is summed up. As a result, the matching positions are highlighted using focus+context visualization. However, performing this process is expected to take a while (e. g., several seconds), hence, it should be available only on demand.

Another enhancement is to make use of the *duality* between the screen space and the (DOI) bin maps, which is illustrated in figure 8.1. All line segments (time series) that pass through a certain pixel of the display in figure (a) are transformed to points that lie on a straight line in the bin map (b), and vice versa. Accordingly, one can apply an approach, which is similar to volume rendering techniques: For each pixel in the final representation, the entries of the associated line in the (DOI) bin map are sampled (see the orange points in Fig. 8.1 (b)). In doing so, one gets the density value and the DOI information of the respective pixel. Using a fragment program and textures representing the (DOI) bin maps, this approach can be efficiently implemented on graphics hardware.

The (DOI) bin maps currently represent a discrete frequency-based representation of the time series. In order to cope with the granularity issue, one can rather "splat" than aggregate the time series into the maps (compare to the work of Westover [90] in the context of volume rendering). In doing so, each time series would affects multiple adjacent 2D bins in the map.

# Acknowledgegements

I want to thank *Helwig Hauser* and *Helmut Doleisch* for their steady support and their valuable feedback and input to the project and this thesis. Special thanks go to *Philipp Muigg*, who supported me in every issue related to the work and the thesis, also for the inspiring discussions we had. Furthermore, I want to thank the *VRVis Center* and my colleagues, namely *Harald Piringer* and *Raphael Fuchs* for their valuable input.

This work has been done in the scope of the *INDICATE project*, which is funded by the Austrian Science Fond *FWF* (Nr. P18733-N10), in collaboration with the *Wegener Center for Climate and Global Change*, University Graz. In this context, I want to thank *Mag. Florian Ladstätter* and *Dr. Andrea Steiner* for their expertise and assistance, furthermore, *Prof. Gottried Kirchengast*, without whom the collaboration in the project INDICATE would not have been established.

The CFD dataset is courtesy of the AVL List GmbH in Graz, Austria. The hurricane data set is provided by the the Mesoscale and Microscale Meteorology (MMM) division at NCAR (National Center for Atmospheric Research) in Boulder, United States. The ERA-40 data set is courtesy of the European Centre for Medium-Range Weather Forecasts (ECMWF) in London, United Kingdom.

Besides work, I want to thank my circle of friends for being there for me, especially *David Horn*, who was so kind to proofread my English, and *Alex Degelsegger*. Finally, I want to thank *my parents* Rudolf and Gertrude Kehrer for everything they have done for me—this thesis is dedicated to them.

# Bibliography


[1] Aigner, W., S. Miksch, W. Müller, H. Schumann and C. Tominski: *Visual Methods for Analyzing Time-Oriented Data*. IEEE Trans. on Visualization and Computer Graphics, 2007. to appear.

[2] Aigner, W., S. Miksch, W. Müller, H. Schumann and C. Tominski: *Visualizing Time-Oriented Data: A Systematic View*. Computers and Graphics, 31(3):401–409, 2007.

[3] Aigner, W., S. Miksch, B. Thurnher and S. Biffl: *PlanningLines: novel glyphs for representing temporal uncertainties and their evaluation*. In *Proc. of the 9th International Conference on Information Visualisation (IV '05)*, pp. 457–463, 2005.

[4] Andrienko, N. and G. Andrienko: *Exploratory Analysis of Spatial and Temporal Data - A Systematic Approach*. Springer, 2006.

[5] Becker, R. A. and W. S. Cleveland: *Brushing scatterplots*. Technometrics, 29(2):127–142, 1987.

[6] Bendix, F., R. Kosara and H. Hauser: *Parallel Sets: Visual Analysis of Categorical Data*. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2005)*, p. 18, Washington, DC, USA, 2005. IEEE Computer Society.

[7] Berry, L. and T. Munzner: *BinX: Dynamic Exploration of Time Series Datasets Across Aggregation Levels*. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2004)*, p. 215.2, Washington, DC, USA, 2004. IEEE Computer Society.

[8] Buja, A., J. McDonald, J. Michalak and W. Stuetzle: *Interactive data visualization using focusing and linking*. In *Proceedings IEEE Visualization '91 (Vis '91)*, pp. 156–163, 1991.

[9] Buono, P., A. Aris, C. Plaisant, A. Khella and B. Shneiderman: *Interactive pattern search in time series*. In *Proc. IST/SPIE's 17th Ann. Intl. Symp. Electronic Imaging (VDA '05)*, vol. 5669, pp. 175–186, 2005.

[10] Buono, P., C. Plaisant, A. Simeone, A. Aris, G. Shmueli and W. Jank: *Similarity-Based Forecasting with Simultaneous Previews: A River Plot Interface for Time Series Forecasting*. In *Proc. of the 11th International Conference on Information Visualisation (IV '07)*, pp. 191–196, Washington, DC, USA, 2007. IEEE Computer Society.

[11] Chan, K. pong and A. W.-C. Fu: *Efficient Time Series Matching by Wavelets*. In *Proc. of the 15th International Conference on Data Engineering (ICDE '99)*, pp. 126–133, Washington, DC, USA, 1999. IEEE Computer Society.

[12] CHEN, H.: *Compound brushing explained*. Information Visualization, 3(2):96–108, 2004.

[13] CHIU, B., E. KEOGH and S. LONARDI: *Probabilistic discovery of time series motifs*. In *Proc. of the 9th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '03)*, pp. 493–498, 2003.

[14] DAASSI, C.: *Techniques d'interaction avec un espace de données temporelles*. PhD thesis, Université Joseph Fourier, France, 2003.

[15] DOLEISCH, H.: *Visual Analysis of Complex Simulation Data using Multiple Heterogenous Views*. PhD thesis, Vienna University of Technology, Austria, 2004.

[16] DOLEISCH, H., M. GASSER and H. HAUSER: *Interactive Feature Specification for Focus+Context Visualization of Complex Simulation Data*. In *Proc. of the 6th Joint IEEE TCVG - EUROGRAPHICS Symposium on Visualization (VisSym 2004)*, pp. 239–248, 2004.

[17] DOLEISCH, H. and H. HAUSER: *Smooth Brushing for Focus+Context Visualization of Simulation Data in 3D*. In *WSCG 2002 Conference Proceedings*, pp. 147–154, 2002.

[18] DOLEISCH, H., M. MAYER, M. GASSER, P. PRIESCHING and H. HAUSER: *Interactive Feature Specification for Simulation Data on Time-Varying Grids*. In *Simulation und Visualisierung 2005 (SimVis 2005), 3-4 März 2005, Magdeburg*, pp. 291–304, 2005.

[19] DOLEISCH, H., 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)*, pp. 91–96, 343, 2004.

[20] DOLEISCH, H., P. MUIGG and H. HAUSER: *Interactive Visual Analysis of Hurricane Isabel with SimVis*. Techn. Rep. TR-VRVis-2004-058, VRVis Research Center, Vienna Austria, 2004. http://www.vrvis.at/simvis/Isabel/.

[21] DOLEISCH, H., G. STONAWSKI and H. HAUSER: *SimVis: Interactive Visual Analysis of Simulation Results*. In *Proc. of the NAFEMS Seminar on Simulation of Complex Flows (CFD)*, 2005.

[22] FALOUTSOS, C., M. RANGANATHAN and Y. MANOLOPOULOS: *Fast subsequence matching in time-series databases*. In *Proc. of the SIGMOD international conference on Management of data (SIGMOD '94)*, pp. 419–429, New York, NY, USA, 1994. ACM Press.

[23] FEKETE, J.-D. and C. PLAISANT: *Interactive Information Visualization of a Million Items*. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pp. 117–124, Washington, DC, USA, 2002. IEEE Computer Society.

[24] FERNANDO, R. and M. J. KILLGARD: *The Cg Tutorial, The Definitive Guide to Programmable Real-Time Graphics*. Addison-Wesley, 2003.

[25] FRANK, A. U.: *Different types of "Times" in GIS*. In EGENHOFER, M. J. and R. G. GOLLEDGE (eds.): *Spatial and temporal reasoning in geographic information systems*, pp. 40–62. Oxford University Press, 1998.

[26] FURNAS, G. W.: *The FISHEYE View: New Look at Structured Files*. Techn. Rep. #81-11221-9, Bell Laboratories, Murray Hill, New Jersey 07974, U.S.A., 1981.

[27] FURNAS, G. W.: *Generalized fisheye views*. In *Proc. of the ACM SIGCHI conference on Human factors in computing systems (CHI '86)*, pp. 16–23, 1986.

[28] HAO, M., D. KEIM, U. DAYAL and T. SCHRECK: *Multi-Resolution Techniques for Visual Exploration of Large Time-Series Data*. In *Eurographics/IEEE-VGTC Symposium on Visualization*, 2007.

[29] HARRIS, R. L.: *Information Graphics: A Comprehensive Illustrated Reference*. Oxford University Press, Inc., New York, NY, USA, 1999.

[30] HAUSER, H.: *Generalizing Focus+Context Visualization*. In BONNEAU, G.-P., T. ERTL and G. M. NIELSON (eds.): *Scientific Visualization: The Visual Extraction of Knowledge from Data (Proc. of the Dagstuhl 2003 Seminar on Visualization)*, pp. 305–327. Springer, 2005.

[31] HAUSER, H.: *Toward new grounds in visualization*. SIGGRAPH Comput. Graph., 39(2):5–8, 2005.

[32] HAUSER, H., F. LEDERMANN and H. DOLEISCH: *Angular brushing of extended parallel coordinates*. In *Proc. IEEE Symposium on Information Visualization 2002 (InfoVis 2002)*, pp. 127–130, 2002.

[33] HAVRE, S., E. HETZLER, P. WHITNEY and L. NOWELL: *ThemeRiver: Visualizing Thematic Changes in Large Document Collections*. IEEE Trans. on Visualization and Computer Graphics, 8(1):9–20, 2002.

[34] HETLAND, M. L.: *A survey of recent methods for efficient retrieval of similar time sequences*. In *Data Mining in Time Series Databases*. World Scientific, 2003.

[35] HOCHHEISER, H.: *Interactive graphical querying of time series and linear sequence data sets*. PhD thesis, University of Maryland, College Park, USA, 2003.

[36] HOCHHEISER, H. and B. SHNEIDERMAN: *Dynamic query tools for time series data sets: timebox widgets for interactive exploration*. Information Visualization, 3(1):1–18, 2004.

[37] JOHANSSON, J., P. LJUNG, M. JERN and M. COOPER: *Revealing structure in visualizations of dense 2D and 3D parallel coordinates*. Information Visualization, 5(2):125–136, 2006.

[38] KEIM, D., W. MUELLER and H. SCHUMANN: *Visual data mining*. In *Eurographics 2002 State-of-the-Art Reports*, pp. 49–68, 2002.

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

[40] KEIM, D. A., F. MANSMANN, J. SCHNEIDEWIND and H. ZIEGLER: *Challenges in Visual Data Analysis*. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2006)*, pp. 9–16, Washington, DC, USA, 2006. IEEE Computer Society.

[41] KEOGH, E.: *Data mining and machine learning of time series data: a tutorial*. Proc.14th European Conference on Machine Learning (ECML) and the 7th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD), 2003.

[42] KEOGH, E. and S. KASETTY: *On the need for time series data mining benchmarks: a survey and empirical demonstration*. In *Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pp. 102–111, New York, NY, USA, 2002. ACM Press.

[43] KEOGH, E., S. LONARDI and B. Y. CHI' CHIU: *Finding surprising patterns in a time series database in linear time and space*. In *Proc. of the 8th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '02)*, pp. 550–556, 2002.

[44] KEOGH, E., S. LONARDI and C. A. RATANAMAHATANA: *Towards parameter-free data mining*. In *Proc. of the 10th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '04)*, pp. 206–215, New York, NY, USA, 2004.

[45] KEOGH, E. and M. PAZZANI: *An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback*. In *Proc. of the 4th ACM SIGKDD international conference on Knowledge discovery and data mining (KDD '98)*, pp. 239–241, New York City, NY, 1998.

[46] KEOGH, E. and C. A. RATANAMAHATANA: *Exact indexing of dynamic time warping*. Knowledge and Information Systems, 7(3):358–386, 2005.

[47] KLEMENT, E. P., R. MESIAR and E. PAP: *Triangular Norms*, vol. 8 of *Trends in Logic*. Kluwer Academic Publishers, 2000.

[48] KONYHA, Z., K. MATKOVIC, D. GRACANIN, M. DURAS, J. JURIC and H. HAUSER: *Interactive Visual Analysis of a Timing Chain Drive Using Segmented Curve View and other Coordinated Views*. In *Proc. of the Intl. Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2007)*, pp. 3–15, Washington, DC, USA, 2007. IEEE Computer Society.

[49] KONYHA, Z., K. MATKOVIC, D. GRACANIN, M. JELOVIC and H. HAUSER: *Interactive Visual Analysis of Families of Function Graphs*. IEEE Trans. on Visualization and Computer Graphics, 12(6):1373–1385, 2006.

[50] KOSARA, R., 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)*, pp. 45–54, 2004.

[51] KURSINSKI, E. R., G. A. HAJJ, J. T. SCHOFIELD, R. P. LINFIELD and K. R. HARDY: *Observing Earth's atmosphere with radio occultation measurements using the Global Positioning System*. Journal of Geophys. Research, 102:23429–23466, 1997.

[52] LADSTÄDTER, F., A. K. STEINER, B. C. LACKNER, G. KIRCHENGAST, P. MUIGG, J. KEHRER and H. DOLEISCH: *SimVis: An Interactive Visual Field Exploration Tool applied to Climate Research*. In *Poster presentation at the 3rd Intern. Workshop on Occultations for Probing Atmosphere and Climate (OPAC-3)*, Graz, 2007.

[53] LARAMEE, R. S., C. GARTH, H. DOLEISCH, J. SCHNEIDER, H. HAUSER and H. HAGEN: *Visual Analysis and Exploration of Fluid Flow in a Cooling Jacket*. In *Proceedings IEEE Visualization 2005 (Vis '2005)*, pp. 623–630, 2005.

[54] LARAMEE, R. S., B. JOBARD and H. HAUSER: *Image Space Based Visualization of Unsteady Flow on Surfaces*. In *Proceedings IEEE Visualization 2003 (Vis '2003)*, pp. 131–138, Washington, DC, USA, 2003. IEEE Computer Society.

[55] LEVOY, M.: *Display of Surfaces from Volume Data*. IEEE Computer Graphics and Applications, 8(3):29–37, 1988.

[56] LIN, J., E. KEOGH and S. LONARDI: *Visualizing and discovering non-trivial patterns in large time series databases*. Information Visualization, 4(2):61–82, 2005.

[57] LIN, J., E. KEOGH, L. WEI and S. LONARDI: *Experiencing SAX: A Novel Symbolic Representation of Time Series*. Data Mining and Knowledge Discovery Journal, to appear, 2007.

[58] LÓPEZ, I. F. V., R. T. SNODGRASS and B. MOON: *Spatiotemporal Aggregate Computation: A Survey*. IEEE Trans. on Knowledge and Data Engineering, 17(2):271–286, 2005.

[59] MACEACHREN, A.: *How Maps Work: representation, visualization, and design*. Guilford Press, New York, 1995.

[60] MARTIN, A. and M. WARD: *High Dimensional Brushing for Interactive Exploration of Multivariate Data*. In *Proceedings IEEE Visualization '95 (Vis '95)*, pp. 271–278, 1995.

[61] MATKOVIC, K., D. GRACANIN, Z. KONYHA and H. HAUSER: *Color LinesView: AnApproach toVisualization of Families of Function Graphs*. In *Proc. IEEE Symposium on Information Visualization 2004 (InfoVis 2007)*, pp. 59–64, Washington, DC, USA, 2007. IEEE Computer Society.

[62] MUIGG, P., M. HADWIGER, H. DOLEISCH and H. HAUSER: *Scalable Hybrid Unstructured and Structured Grid Raycasting*. In *Proceedings IEEE Visualization 2007 (Vis '2007)*, 2007. to appear.

[63] MÜLLER, W., T. NOCKE and H. SCHUMANN: *Enhancing the visualization process with principal component analysis to support the exploration of trends*. In *Proc. of the Asia Pacific symposium on Information visualisation (APVIS '06)*, pp. 121–130, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[64] MÜLLER, W. and H. SCHUMANN: *Visualization methods for time-dependent data - an overview*. In *Proc. of the 35th conference on Winter simulation*, pp. 737–745, 2003.

[65] NOCKE, T., H. SCHUMANN and U. BÖHM: *Methods for the Visualization of Clustered Climate Data*. Computational Statistics, 19(1):75–94, 2004.

[66] NOCKE, T., H. SCHUMANN, U. BÖHM and M. FLECHSIG: *Information Visualization Supporting Modelling and Evaluation Tasks for Climate Models*. In *Proc. of the 35th conference on Winter simulation*, pp. 763–771, 2003.

[67] NOVOTNY, M. and H. HAUSER: *Outlier-Preserving Focus+Context Visualization in Parallel Coordinates*. IEEE Trans. on Visualization and Computer Graphics, 12(5):893–900, 2006.

[68] PIRINGER, H., R. KOSARA and H. HAUSER: *Interactive Focus+Context Visualization with Linked 2D/3D Scatterplots*. In *Proc. of the Intl. Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2004)*, pp. 49–60, 2004.

[69] POST, F. H., B. VROLIJK, H. HAUSER, R. S. LARAMEE and H. DOLEISCH: *Feature Extraction and Visualization of Flow Fields*. In *Eurographics 2002 State-of-the-Art Reports*, pp. 69–100, 2002.

[70] POST, F. H., B. VROLIJK, H. HAUSER, R. S. LARAMEE and H. DOLEISCH: *The State of the Art in Flow Visualisation: Feature Extraction and Tracking*. Comput. Graph. Forum, 22(4):775–792, 2003.

[71] RATANAMAHATANA, C. A. and E. KEOGH: *Making Time-series Classification More Accurate Using Learned Constraints*. In *Proc. of SIAM International Conference on Data Mining (SDM '04)*, pp. 11–22, 2004.

[72] REINDERS, F., F. H. POST and H. J. W. SPOELDER: *Visualization of time-dependent data with feature tracking and event detection*. The Visual Computer, 17(1):55–71, 2001.

[73] ROBERTS, J. C.: *Exploratory Visualization with Multiple Linked Views*. In MACEACHREN, A., M.-J. KRAAK and J. DYKES (eds.): *Exploring Geovisualization*, pp. 159–180. Elseviers, 2004.

[74] ROBERTS, J. C.: *State of the Art: Coordinated & Multiple Views in Exploratory Visualization*. In ANDRIENKO, G., J. C. ROBERTS and C. WEAVER (eds.): *Proc. of the Intl. Conference on Coordinated & Multiple Views in Exploratory Visualization (CMV 2007)*, pp. 61–71. IEEE Computer Society Press, 2007.

[75] ROBERTS, J. C. and M. A. E. WRIGHT: *Towards Ubiquitous Brushing for Information Visualization.*. In *Proc. of the 10th International Conference on Information Visualisation (IV '06)*, pp. 151–156. IEEE Computer Society, 2006.

[76] RODDICK, J. F. and M. SPILIOPOULOU: *A Survey of Temporal Knowledge Discovery Paradigms and Methods*. IEEE Trans. on Knowledge and Data Engineering, 14(4):750–767, 2002.

[77] Ryall, K., N. Lesh, T. Lanning, D. Leigh, H. Miyashita and S. Makino: *QueryLines: approximate query for visual browsing*. In *Extended abstracts on Human factors in computing systems (CHI '05)*, pp. 1765–1768, New York, NY, USA, 2005. ACM Press.

[78] Schumann, H. and W. Müller: *Visualisierung – Grundlagen und allgemeine Methoden*. Springer, 2000. in German.

[79] Shneiderman, B.: *The Eyes Have It: A Task by Data Type Taxonomy for Information Visualizations*. In *IEEE Visual Languages*, pp. 336–343, College Park, Maryland 20742, U.S.A., 1996.

[80] Silva, S. F. and T. Catarci: *Visualization of Linear Time-Oriented Data: A Survey*. In *WISE '00: Proceedings of the First International Conference on Web Information Systems Engineering (WISE'00)-Volume 1*, p. 310, Washington, DC, USA, 2000. IEEE Computer Society.

[81] Stacey, M. and C. McGregor: *Temporal abstraction in intelligent clinical data analysis: A survey*. Artif. Intell. Med., 39(1):1–24, 2007.

[82] Thomas, J. and K. Cook: *Illuminating the Path: Research and Development Agenda for Visual Analytics*. IEEE-Press, 2005.

[83] Thomas, J. and K. Cook: *A visual analytics agenda*. IEEE Computer Graphics and Applications, 26(1):10–13, 2006.

[84] Tominski, C.: *Event-Based Visualization for User-Centered Visual Analysis*. PhD thesis, Institute for Computer Science, Department of Computer Science and Electrical Engineering, University of Rostock, 2006.

[85] Tominski, C., J. Abello and H. Schumann: *Axes-based visualizations with radial layouts*. In *Proc. of the ACM Symposium on Applied Computing (SAC '04)*, pp. 1242–1247, New York, NY, USA, 2004.

[86] Tominski, C., P. Schulze-Wollgast and H. Schumann: *3D Information Visualization for Time Dependent Data on Maps*. In *Proc. of the 9th International Conference on Information Visualisation (IV '05)*, pp. 175–181. IEEE Computer Society, 2005.

[87] Tufte, E. R.: *The Visual Display of Quantitative Information*. Graphics Press, Cheshire, Connecticut, USA, 1983.

[88] Wattenberg, M.: *Sketching a graph to query a time-series database*. In *Extended abstracts on Human factors in computing systems (CHI '01)*, pp. 381–382. ACM Press, 2001.

[89] Wegman, E. J. and Q. Luo: *High dimensional clustering using parallel coordinates and the grand tour*. Computing Science and Statistics, 28:352–360, 1997.

[90] WESTOVER, L.: *Footprint evaluation for volume rendering*. In *Proc. of the 17th annual conference on Computer graphics and interactive techniques (SIGGRAPH '90)*, pp. 367–376, New York, NY, USA, 1990. ACM Press.

[91] WIJK, J. J. VAN: *Image based flow visualization*. In *Proc. of the 29th annual conference on Computer graphics and interactive techniques (SIGGRAPH '02)*, pp. 745–754, New York, NY, USA, 2002. ACM Press.

[92] WIJK, J. J. VAN: *The value of visualization*. In *Proceedings IEEE Visualization 2005 (Vis '2005)*, pp. 79–86, 2005.

[93] WIJK, J. J. VAN and E. R. V. SELOW: *Cluster and Calendar Based Visualization of Time Series Data*. In *Proc. IEEE Symposium on Information Visualization 1999 (InfoVis '99)*, pp. 4–9, 1999.

[94] WILLS, G.: *Selection: 524,288 ways to say "this is interesting"*. In *Proc. IEEE Symposium on Information Visualization 1996 (InfoVis '96)*, pp. 54–61, 1996.

[95] XU, R. and D. WUNSCH: *Survey of clustering algorithms*. IEEE Trans. on Neural Networks, 16(3):645–678, 2005.

[96] ZADEH, L.: *Fuzzy sets*. Information and Control, 8:338–353, 1965.