Interactive Visual Analysis of Streaming Data

by

Geir Smestad

Master Thesis in Visualization

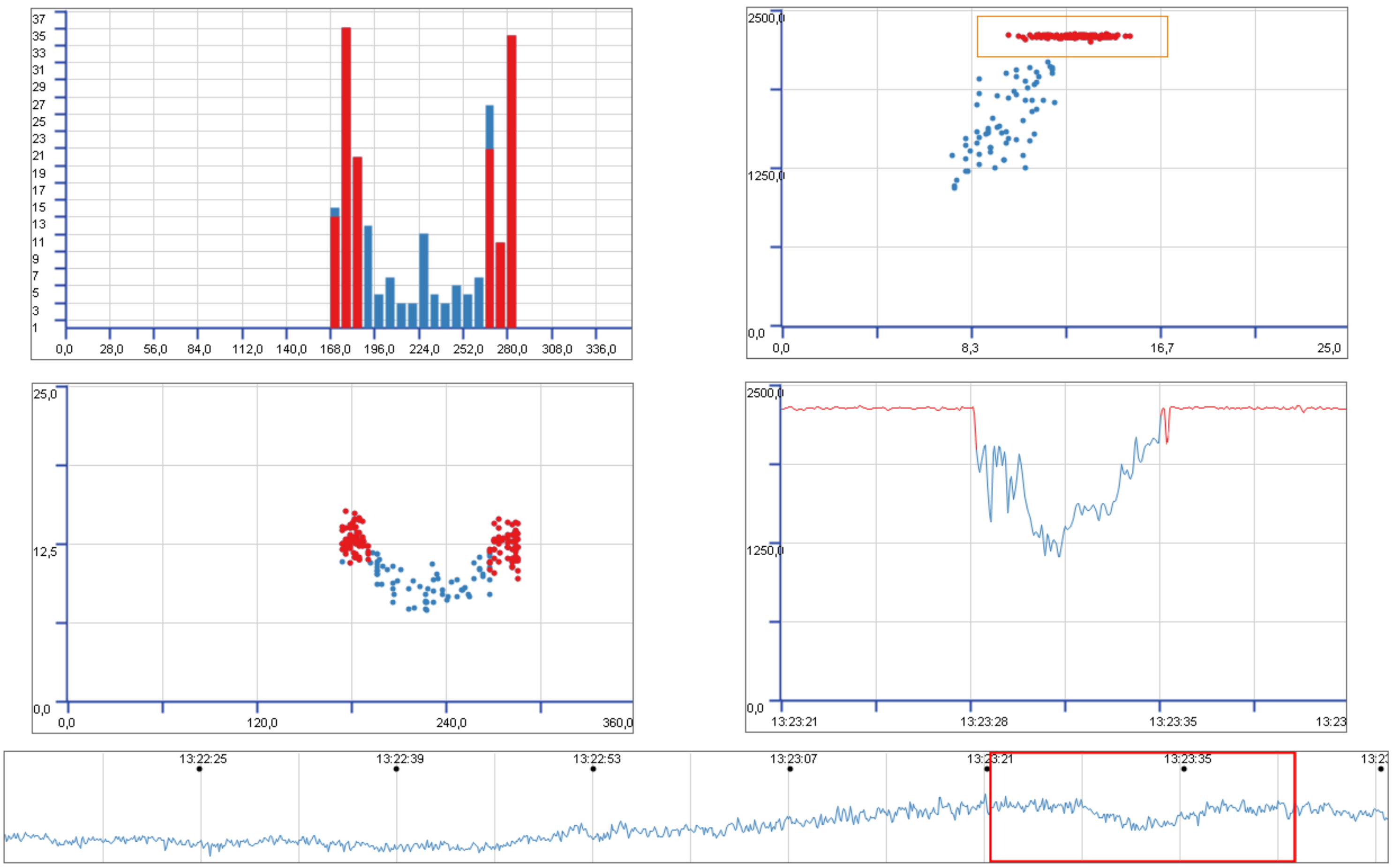
Institute of Informatics
University of Bergen
Norway

August 2014

Interactive Visual Analysis of Streaming Data

Geir Smestad

geir.smestad@gmail.com



Supervised by Professor Helwig Hauser

Institute of Informatics
Visualization Group
University of Bergen

August 2014

http://www.ii.uib.no/vis/teaching/thesis/2014-Smestad/

Abstract

Interactive Visual Analysis (IVA) has proven to be a robust set of methods for visually exploring complex data sets and generating hypotheses from data. Datasets and techniques where the temporal aspect is central has been an important area of study, both for the visualization field in general and for research on IVA. However, the challenge of handling *streaming* data sources for the purposes of decision support and analysis in *real time*, has been given comparatively little attention.

This thesis presents a summary of the visualization literature addressing time-oriented and streaming data, with emphasis on Interactive Visual Analysis and its related techniques. We then explain the contemporary distinction between real-time data *monitoring* and retrospective data *analysis*, explore challenges that occur when a human user attempts to visually analyze data in real time, and use these observations to extend the scope of IVA such that it can be used to analyze streaming data in real time.

Acknowledgements

First and foremost, I extend my thanks to my brilliant thesis advisor, Professor Helwig Hauser. Helwig's excellent ideas, insights and mastery of both the visualization field and the scientific discipline have been key to the success of this project. My thesis would not have been possible without his many important contributions, and I am deeply grateful for his open and honest feedback. Having a good thesis advisor is key when studying for a higher degree, and I could simply not have wished for a better guide and work partner than Helwig. The VisGroup lunches and everyone in the research group have also been a big inspiration and source of support for the last two years.

Further, thanks to Dr. Ove Daae Lampe and the research community at Christian Michelsen Research in Bergen for their industry contacts and ideas for applying visualization research to industrial applications. Essential credit also goes to Georg Oftedal, Dr. Nenad Keseric and the rest of the Wind Operations Strategy and Support division at Statoil. Their ideas, trust and exceptional openness regarding both data access and insight into their everyday work challenges, have been a big inspiration for this thesis. I greatly value our exchange of knowledge and ideas, and consider it an important contribution to my studies. This type of open and honest exchange between industry and academia is by no means guaranteed.

I also extend my greatest thanks to my sister Ingrid and my parents Eli Marie and Egil, for their continuing company and moral support in things big and small. This also goes to all of my friends, whose company I greatly enjoy. Thanks also to the skuglekugle Computer Scientist community for their general recreational inspiration.

Finally, my deepest thanks to Carol, my beautiful and brilliant girlfriend. She is a wonderful source of love, joy and inspiration.

Contents

List of Figures

List of Videos

If any of the videos on Vimeo are password protected, please use the password ”visgroup”.

Examples

Fast forward: http://vimeo.com/geirsmestad/iva-fastforward

Brushing modes: http://vimeo.com/geirsmestad/iva-brushing

Intermittent updates: http://vimeo.com/geirsmestad/iva-intermittent

Automatic zoom: http://vimeo.com/geirsmestad/iva-autozoom

Histograms: http://vimeo.com/geirsmestad/iva-histograms

Age emphasis in histogram and scatterplot: http://vimeo.com/geirsmestad/iva-ageemphasis

Demonstration scenarios

Demo scenario 1: http://vimeo.com/geirsmestad/iva-general

Demo scenario 2: http://vimeo.com/geirsmestad/iva-quickchanges

Demo scenario 3: http://vimeo.com/geirsmestad/iva-archivesandlensing

Demo scenario 4: http://vimeo.com/geirsmestad/iva-eventsandcomparison

Demo scenario 5: http://vimeo.com/geirsmestad/iva-zoomandintermittent

Redundant video URLs on the UiB server

Fast forward: http://www.ii.uib.no/~geirsm/sd-iva/FastForward.mp4

Abbreviations

2TPC	Two-tone pseudo coloring
API	Application Programming Interface
CMV	Coordinated Multiple Views
CSV	Comma-separated values
CT	Computed Tomography
EBO	Enhanced Buffer Overview
FIFO	First in, first out
GUI	Graphical User Interface
InfoViz	Information visualization
IVA	Interactive Visual Analysis
KDE	Kernel Density Estimation
PAA	Piecewise Aggregate Approximation
RT	Real-Time
SMV	Simple moving variance
T1	Turbine 1
T2	Turbine 2
UX	User Experience

Chapter 1

Introduction

Throughout the last decades, the computing field has experienced a consistent trend where the amounts of available stored data have grown by orders of magnitude. This trend spans a multitude of domains, from science and engineering to health care, banking, retail, social networks and many others [60]. Moore's law and the increasingly cheap storage space for large amounts of data, provides the background for this development [80][1]. However, while the capability to gather and store data has grown rapidly, our ability to analyze and make sense of this data has grown at a much slower pace [39]. Humans are not good at manually reading and interpreting very large spreadsheets. This situation presents great challenges and opportunities.

Visualization has proved to be an invaluable tool when facing the challenge of large-scale data analysis. When a data analysis process can not be automated entirely, there must be a human in the loop to apply domain knowledge and intuition to the process. The visual sense is strongly developed and often provides an effective and efficient pathway from data to cognition. Visualization is a broad field with a large and growing body of published work, spanning techniques to explore, analyze and present data from a large number of domains.

This thesis explores a segment of the visualization field where it is not necessarily a high data volume which poses the biggest challenge, but the *urgency* of making sense of the data: Namely, the **real-time** visualization and analysis of **live, streaming data**.

[1]Section 1.1, page 4

1.1 Background

There are multiple established methodologies for data analysis where visualization is central. The field of *visual analytics* provides a large number of tools for "analytical reasoning facilitated by interactive visual interfaces" [74], and encompasses many data types and applications. *Interactive Visual Analysis* is a successful, generalized approach for visual interaction with the investigation of scientific[2] data [64]. These methods provide a good framework for using the human visual sense to investigate and analyze complex datasets, where the processing power of computers is combined with the human abilities of abstract reasoning and pattern recognition. Where such tools can be fully utilized, they provide great economic and scientific benefits in the form of knowledge discovery, hypothesis testing and verification, efficiency improvements, better dissemination and presentation, and more.

1.2 The challenges of streaming data

One aspect of these visual analysis methods which has not yet been given comprehensive treatment, is how to relate to real-time situations: Scenarios where the data arrives continuously and there is high urgency in understanding the data and making decisions. While *time* is given comparably comprehensive treatment in the literature, the *real-time* and *streaming* aspects of the visual analysis scenarios are given much less attention. Mansmann et al. point out that "Since most analysis and visualization methods focus on static data sets, adding a dynamic component to the data source results in major challenges for both the automated and visual analysis methods" [55].

Treating the data source as streaming and introducing time limits for the conclusions of the analysis, results in a large number of new challenges. These challenges are both technical and human in nature. The restrictions of time as a physical dimension provide some of these challenges, while the human aspects of interaction, attention and multi-tasking represent others. This is a broad subject which would have a large number of application-specific concerns in an end-user production environment, and we necessarily cover only part of the subject in detail. Nevertheless, we attempt to cover most of the

[2]The word "scientific" refers to the data model described in Chapter 2, not the origin of the data. The scientific data model can also be of great use in technical or other non-academic environments.

general areas of focus that are shared between all streaming data analysis scenarios. Since human factors beyond the visual sense are so important in this process, we devote more attention to interaction techniques than to creating novel visualization mappings. We also investigate to which degree such a real-time approach generalizes to situations where there is less urgency for finding insight and making decisions.

1.3 Objective and scope

Interactive Visual Analysis (IVA) provides a solid, practical framework for visual data analysis, by integrating human insight and pattern recognition with the processing power of computers [64]. We discuss this methodology in detail in the next chapter. Using the techniques and terminology of IVA, we

- Explore the unique complications that occur when performing interactive visual data analysis on streaming data, as opposed to static data.
- Investigate how the various aspects of IVA work in a real-time environment of streaming data, and extend these where required. In situations where new human or technical challenges show up, we suggest techniques to alleviate them.
- Examine whether these techniques do a better job of providing timely insight into streaming data than the current state of the art, and whether there are additional benefits to such an approach.

Successfully developing techniques for the real-time visual analysis of streaming data, enables a new mode of operation regarding how analysts and operators treat streaming data. As we discuss in the next chapter, there is currently a distinction between *data analysis* and *data monitoring*. In the former case, the analyst has enough time to investigate complex relationships in the data, but at the expense of not having access to the data immediately after it becomes available. In the latter case, the operator is monitoring the data as it arrives, but has a reduced ability to investigate higher-dimensional correlations in the data. It is our hypothesis that these scenarios can to a certain degree be merged, increasing the limit for the complexity of processes that can be manually monitored and steered in real time.

1.4 A note on terminology

This thesis is about visualizing and analyzing data that arrives in front of an operator concurrently with the operator's analysis procedure. This is a scenario which has in the literature only implicitly or to a limited degree been distinguished from the scenario where the data source is static and separated from the analysis procedure. This means that the terminology for this subject is not yet rigidly defined, so we spend a few moments to ensure that usage of terms is clear.

In visualization in general, the term **real-time** refers to visualizations that are responsive; respond immediately to user input and are updated at animated frame rates. This is a major concern also with static data sources, e.g. in medical 3D volume rendering where large computational resources are required to generate images or animations, and interactivity is paramount. The term **streaming data** refers to situations where the *data arrives continuously*[3]. Where the distinction is required, I always use the latter.

However, there is a degree of overlap between these terms, since a visualization of streaming data will in most cases also be real-time (i.e. responsive, with interactive frame rates). In some cases, it also makes sense to use the term *real-time* or *real time* when we are discussing actions that take place in immediate response to an event. The reader should keep this in mind when reading the text[4].

The words *user*, *operator* and *analyst* are used interchangeably to refer to the person that uses a visualization system, for the purpose of understanding a data stream and taking action based on the observations.

1.5 About the data

In order to exemplify the techniques developed throughout this thesis, we consider sensor data from Statoil's *Hywind* offshore wind turbine [4]. This dataset includes power produced (in kilowatts), wind speed, wind direction, angle of the turbine blades (in degrees)

[3]The term *continuous* should be understood in the practical sense and not by its mathematical definition. Continuously arriving data are of course at some level discrete, as long as they are processed with a digital computer.

[4]Note that in the existing literature, there is also sometimes some ambiguity between these terms. There are examples where the term *real-time* is used to refer to situations where the data source is updated continuously, contrasting with the definitions we use here. Sometimes, the word *dynamic* is used about such visualizations and data sources.

and other parameters of interest. We also use a purely synthetic dataset to exemplify some of our techniques, since synthetic data sometimes provide a clearer illustration of novel visualization techniques. In the demonstration chapter, a larger dataset from Statoil and Statkraft's 88-turbine *Sheringham Shoal* offshore wind farm is used to give an indication on how our techniques would work in a more complex scenario [5].

Both of the test data sets have a resolution of one minute per data point. This means that our data don't represent an ideal example of streaming data in an urgent analysis situation. However, "playing back this data at higher speed" will still provide a test scenario where our techniques can be tested with an acceptable degree of accuracy. It proved difficult to find data that exactly matched our objective, and the exceptionally open attitude of the Statoil Wind Operations group provided us with a solid understanding of these datasets. We consider this a very reasonable tradeoff.

1.6 Video clips

The thesis is also supplemented with a number of video clips that illustrate some of its concepts. These are listed in the *List of Videos*, each with a title and a URL where it can be viewed. The videos are separated into two sections: Six *examples* of our techniques for illustrative purposes, and five *demonstration* scenarios that demonstrate our techniques on real data in a simulated analysis scenario. The videos are introduced throughout the text.

The videos are available from two redundant locations. The main repository is the online video service Vimeo [54]. Since some of the videos are in HD quality and must be compressed for streaming over the web, redundant links to all videos are provided from the UiB Visualization Group server in case the Vimeo links are unavailable or low quality. For the five *demonstration* videos, an HD monitor is required to view the videos in full detail. These videos demonstrate our completed software prototype in practice.

If any of the videos on Vimeo are password protected, please use the password "visgroup".

1.7 Thesis structure

In the following chapter, we provide a review of the scientific context of this thesis, mostly in terms of published, related work: Motivation, techniques and terminology – both from the perspective of analyzing time-oriented data in general, and for streaming data and real-time scenarios in particular.

Chapter 3 presents our main contribution. We provide a broad overview of the challenges of streaming data analysis. We define a visualization framework suitable for the subsequent discussion of the challenges and extensions to streaming data IVA. We develop new interaction methods that increase the operator's time management capabilities, create higher-level transformation operations that can be applied to streaming data and investigate some new visualization techniques suitable for this domain.

Chapter 4 evaluates the techniques developed throughout Chapter 3 through an interview with the domain experts at Statoil's Wind Operations group. This evaluation is followed by five demonstration scenarios, which demonstrate our techniques on real data. The demonstration scenarios are recorded on video, and are supplemented by a comprehensive textual description.

We then summarize our implementation of the visualization framework described in Chapter 3, including the programming environment, data model, GUI frameworks and other architectural choices.

The last two chapters summarize our findings and the lessons learned throughout the project, and suggest directions for future research on this topic.

Chapter 2

Scientific context and related work

This chapter introduces the scientific background for this thesis. This includes a bird's-eye overview of the visualization discipline and an introduction to information visualization and its approaches. We discuss the state of the art in real-time and streaming data visualization, existing techniques to visualize *time-oriented data* and other techniques that are useful for the visual analysis of streaming data.

2.1 Visualization

Visualization is the art and science of creating visual representations of information, with the representations usually being interactive. The rationale for this field is that the human visual sense is strongly developed, as millions of years of evolution have shaped our cognitive abilities to reason, based on the perceivable portion of the electromagnetic rays in our environment. Hence, well-designed visualizations exploit our natural ability to explore and draw conclusions based on visual observations, even if the information that is visualized has no *natural* visual representation. Visualization helps us to observe invisible processes, absorb large information quantities quickly and understand and reason about abstract data. Many of these points were already mentioned by McCormick et al. in their 1987 "Visualization in Scientific Computing" report, which has been instrumental for forming visualization as its own scientific discipline [61].

In general, visualization from the perspective of science and engineering serves three distinct purposes [70]:

- **Exploration**: Finding unknown or unexpected correlations, generating hypotheses
- **Analysis**: Verifying or disproving hypotheses. Finding more complex correlations.
- **Presentation**: Communicating insight to collaborators or decision makers.

Our focus is primarily on exploration and analysis.

2.1.1 Information visualization

A subfield of visualization, *information visualization* (InfoViz) addresses abstract visualization techniques, often applied to data that has no natural, visual representation in the physical world. One example of a successful InfoViz technique is the scatterplot, illustrated in figure 2.1. Data visualized through InfoViz are often of an abstract nature, e.g. census information and statistics, stock market trades or survey results, or physical but invisible properties, such as temperature or pressure in a simulation or a sensor network. This is in contrast to e.g. rendering techniques for data from medical scanners such as CT or ultrasound, where the data has a concrete spatial interpretation. There is a degree of overlap between the different fields of visualization, so this distinction is not strict [27]. Ben Shneiderman coined the Visual Information Seeking Mantra: ”Overview first, zoom and filter, then details-on-demand” as an observation of a typical pattern for designing interaction techniques for this type of data [71].

In information visualization, the data often has (though is not limited to) a ”spreadsheet-like” format where the rows represent the individual measurements and the columns represent different parameters (e.g. temperature, location or income) of each individual measurement. This data format generalizes to a wide range of applications. Once a data source is defined, information visualization introduces methods to query, analyze, interpret, understand and convey the information that is contained in the observations. Techniques in information visualization are widely used in science, engineering and the media to convey and illustrate abstract information.

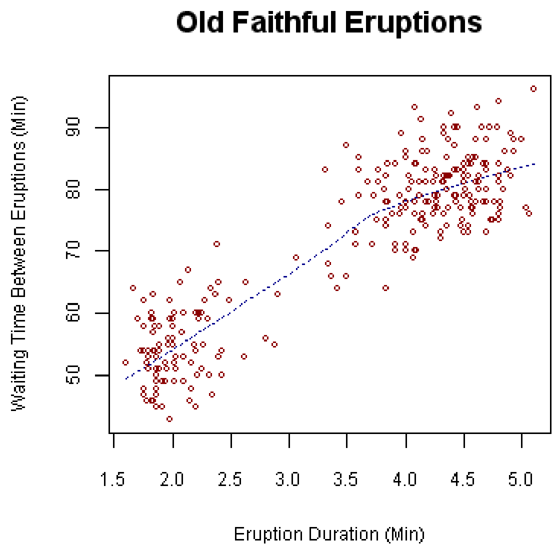


FIGURE 2.1: Scatterplot representing eruptions of the Old Faithful Geyser. Each circle represents one eruption. This plot packs a lot of information in a small amount of space. A human analyst will easily determine that the eruptions are generally grouped in two categories, a pattern recognition feat which would be harder to accomplish using mathematical clustering or just looking at the raw numbers. Image from the Wikimedia Commons [12].

2.2 Interactive Visual Analysis

Interactive Visual Analysis (IVA) [16, 17, 78] can be considered an intersection and generalization of different methods from information visualization and data analysis, representing a generalized set of techniques for combining the computational power of computers with the perceptive and cognitive capabilities of a human analyst [64]. The approach is based on an interactive, iterative visual process where the analyst gradually closes in on the most interesting features of the data. It has proven to be a successful methodology for exploring and analyzing data and generating hypotheses [42]. IVA has been employed in a wide number of scientific disciplines, including engineering [59], medicine [3], climatology [37] and biology [75]. Lampe has explored IVA in the context of process data analysis [47]. While their main focus is the exploration and analysis of data, the techniques employed in IVA can also be used for presenting observations about complex datasets to people who are not themselves intimately familiar with the subject under scrutiny.

2.2.1 Data model

Typically, data which is suitable for being analyzed with IVA is compatible with a spreadsheet-like representation, is multi-dimensional (several or even many columns) and encompasses many data points (many rows). More generally, the data model consists of *independent variables* (domain) such as space or time, and *dependent variables* (range) such as pressure or temperature. For a given observation, we can consider the dependent variables with respect to the independent variables as $\mathbf{d}(\mathbf{x})$, where $\mathbf{x} \in \mathbf{domain}$ and $\mathbf{d} \in \mathbf{range}$. The dimensionalities of the domain and range vary based on the application, but are in the context of IVA sufficiently high so that the entire dataset can not be usefully displayed as a whole. This data model is known in some texts as the *scientific data model* [42].

An example of a dataset following this data model could be a log of sensor data from a wind turbine, containing second-to-second updates of wind speed, wind direction, blade pitch, electrical power output and other variables which the analyst suspects are correlated. In this case, *time* and *turbine number* or *position* would be the independent variables, and all other variables would be dependent on them. IVA could be used to explore such a dataset, detect anomalies and perhaps discover ways in which the turbine operation could be made more economical.

2.2.2 Linked views

At the heart of IVA is the concept of Coordinated Multiple Views (CMV) [23, 67]. At least two different views are displayed, each showing a particular representation of the data points in the dataset. These views could use visualization techniques from scientific visualization, e.g. volume rendering, or techniques from information visualization such as scatterplots, histograms or parallel coordinates [42].

A key feature of CMV is that the views are linked, which means that a data point highlighted (”brushed”) by the user in one view will be automatically and consistently highlighted in all the other views. This is a very intuitive method of viewing and exploring higher-dimensional relationships in a dataset. This ”brushing and linking” [42] is tightly related to the important visualization concept of ”Focus+Context”, where the selected data points corresepond to the current focus of the analysis and the surrounding

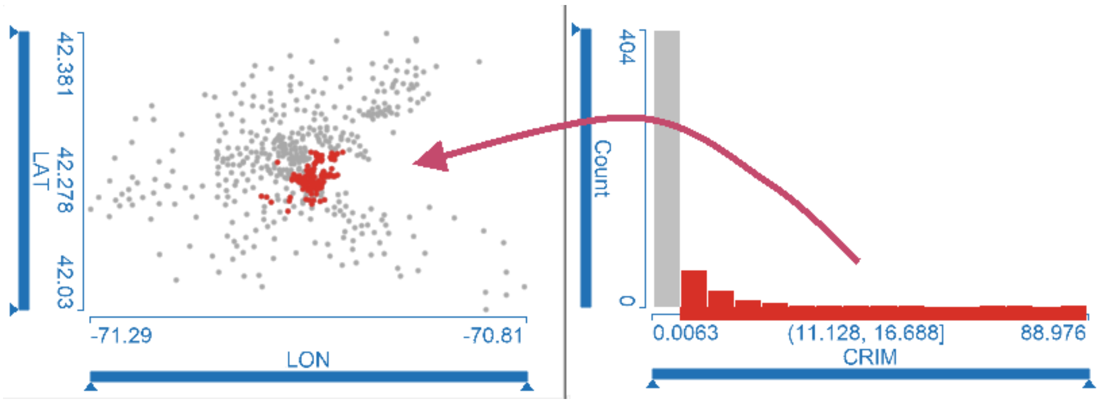


FIGURE 2.2: Brushing in coordinated, linked views. Illustration created with ComVis [58]. This the well-known 1978 Boston Housing Dataset [25]. The scatterplot on the left shows the location of each sample in the Boston metropolitan area, the histogram on the right shows the corresponding crime rates. A *feature localization* brush (explained below) of significant crime rates clearly indicates that reported crime is concentrated in the downtown area.

data points are the context of the exploration [28]. Focus+Context initially referred to the explicit simulation of a fisheye lens or a magnifying glass [68], but has later been generalized to a more abstract visualization technique [28].

The concept of brushing can be expanded to smooth brushing, which implies a variable degree of interest in the data points [15]. This improves the Focus+Context aspect of linked views by allowing non-binary selections of data points. For example, when exploring temperature data in a simulation dataset, we might be interested in "high temperatures" in general, not only temperatures above a sharp cut-off threshold like 137 °C. In this case, the brushed selection will be colored with a gradient rather than a sharp distinction in color.

2.2.3 The "Show & brush" loop – first-level IVA

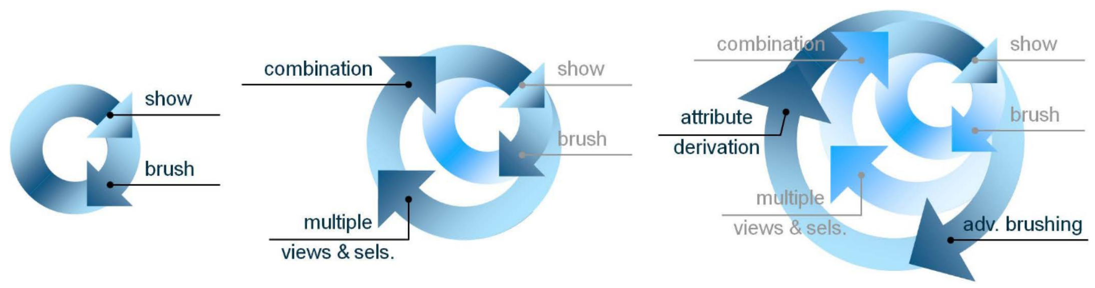


FIGURE 2.3: The Show & brush loop, illustrated for three different levels of complexity (described below). Image courtesy of Konyha et al. [43]

Once some initial linked views are established, the user engages in a "dialog" with the visualization system. The user highlights data points of particular interest in one of the linked views, and the corresponding data points are automatically highlighted in the other views. This ideally leads to an observation or some insight about the dataset. Observations made after brushing in one view will then be used as a basis for brushing again, perhaps in a different view. This is called the "Show & brush loop" [43]. The user may also bring up additional, different views to explore new questions that show up during this process [64].

The Show & brush loop makes IVA more dynamic and intuitive than traditional statistical techniques, which makes it suitable for situations where the user isn't certain about what he or she is looking for. The Show & brush technique is also a *much* faster way to make successive queries of the data than traditional analysis techniques [64]. The visual and intuitive aspects of the technique improve the ability to detect and eliminate subtle data errors, ask quick follow-up questions and apply intuition and domain knowledge.

Patterns of IVA

Given a data model with dependent and independent variables that are plotted in the linked views, three different brushing patterns are possible, as described by Oeltze et al. [63]:

1. **Feature localization**: Dependent variables (e.g. temperature) are brushed in order to see which independent variables (e.g. location) correspond to the selected feature. Figure 2.2 shows an example of this pattern.
2. **Local investigation**: Independent variables (e.g. time) are brushed in order to see which dependent variables (e.g. temperature) correspond to the selected range of independent variables (e.g. time span)
3. **Multivariate analysis**: Dependent variables (e.g. temperature) are brushed in order to see how features of another dependent variable (e.g. pressure) correlate, or not.

2.2.4 Advanced IVA techniques

The form of IVA described so far, called *first-level IVA*, has considerable power for hypothesis generation. But the technique can be expanded to be suitable for the discovery of relationships or structures in the data that are more subtle, and hard to detect using a simple approach. Konyha et al. define three levels of complexity [43], with Oeltze et al. defining a fourth [64]:

Second-level IVA

The user might want to create multiple views, or use more than just one brush. By allowing for multiple concurrent brushes on the same dataset [56], with the selection defined as a Boolean combination of these brushes, the potential for information drill-down is significantly increased [16]. This represents the second level of complexity in IVA.

Third-level IVA

Sometimes, the user is interested in deriving attributes from the original dataset, e.g. by calculating a new column in the dataset based on other columns. A canonical example is calculating an estimate of the first derivatives of the X-ray absorption with regards to location in a computed tomography (CT) dataset. These *gradients* can characterize *boundaries* between different regions of the volume, and hence indicate regions of interest, in addition to being an aid during volume rendering [18, 49].

Derived attributes immediately become available to the Show & brush loop. Once suitable attribute derivations have been performed, the user can therefore make queries about more complex semantic aspects of the dataset. In the case we just described, one example could be a query involving the different boundaries and contiguous regions of the volume. If we have a volume rendering of a CT volume (representing a human skull, for instance) and a histogram of the absolute values of the gradients in the dataset, highlighting the largest gradients in the histogram would cause the volume rendering to highlight all sharp boundary regions in the volume. Visually highlighting these areas is hard to do using only the initial data set, but easy to do with attribute derivation. Which attribute derivations to use in practice, is dependent on the problem domain.

An alternative to explicitly deriving attributes is to introduce advanced brushing techniques: The user does not select data points explicitly but rather through a derived measure such as density, average value or first derivative. The options for such brushes are unbounded and only limited by the analyst's willingness to adopt new methods [43].

Fourth-level IVA

The capabilities of IVA can be further expanded by application-specific feature detection. This is only limited by the analyst's imagination, domain knowledge and available time. One example would be detecting vortices in a fluid simulation dataset and assigning a measure of this to each data point, perhaps by doing calculations on a different but related dataset [9].

2.2.5 The role of IVA when exploring complex datasets

As indicated above, Interactive Visual Analysis can be used to uncover hidden information in complex datasets, which is the same objective as addressed by data mining techniques[1]. In contrast to many mathematical and statistical data mining techniques, the interactivity of IVA enables the human operator to very quickly apply intuition and domain knowledge to the data analysis process. In addition, outliers and anomalies in the data are more likely to be discovered than if purely automated processing was performed; these are often easily visually distinguished from the rest of the data. Visual methods also improve the ability to discover unexpected characteristics of the data, and the generality of IVA ensures that it can be applied to a large number of domains.

Some form of data visualization is usually essential if we want to extract knowledge from complex datasets. Just looking at numbers in a spreadsheet or other data structure will fail for all but the simplest data analysis problems. IVA takes advantage of this by bringing the visual process to the center of the analysis methodology. It also serves as a starting point for generating firm statistical measures for a given phenomenon.

[1]The term *data mining* in the scientific literature often refers to specific computational, mathematical or statistical techniques for uncovering information and knowledge from complex datasets. Data mining uses techniques from machine learning in order to discover patterns in data [80] (chapter 1, pages 5–6). The term "data mining" has been adopted by the industry and hence, in the context of business, it is often used as a *general* term for complicated techniques that aid knowledge discovery in complex datasets. When reading the scientific literature with a layman's understanding of the term, the more specific definition of data mining can be a source of confusion.

Statistical analysis is a widely established scientific technique, but it is often not clear when exploring a dataset which parts of the data are relevant for the statistical analysis. Further, it is harder to formulate hypotheses when using statistics on an otherwise unfamiliar dataset. And since the properties of the data are unknown, choosing the right statistical approach is difficult. IVA alleviates all these problems.

2.2.6 Temporal considerations

While data explored with IVA is often also *time-dependent*, the data does not usually arrive to the visualization interface in real time. Most existing papers on IVA either use static datasets, or implicitly assume that the data source is static and does not change during the analysis procedure. This is often a reasonable assumption, and also a useful simplification when developing advanced visualization methods. However, no aspect of IVA *requires* the dataset to be static, which implies that the methodology could be adapted to streaming data. We explore the implications of performing IVA of dynamic, streaming data in the next chapter. This includes a discussion of the applicability of and the adaptions to the advanced IVA techniques as summarized above.

Note that a visualization technique can be interactive and at *real-time* framerates, even if the data itself does not arrive in real time. This distinction can be a source of confusion when reading the literature, as mentioned in the introduction.

2.3 Streaming data visualization

Szewczyk defines streaming data as ”a continuous sequence of ordered observations of indeterminate length” [73]. As discussed by Lampe [47], this has a number of implications when referring to such a data source:

- The data arrives in parallel with the operator’s monitoring or analysis procedure
- The sample size is potentially unbounded
- Data which is not explicitly stored, is irrevocably lost

Although the latter two of these restrictions can usually (though not always) be controlled through suitable memory management techniques, there is a clear implication that creating visualizations of streaming data means managing greater complexity. This is a challenge both from a human and a technical viewpoint.

2.3.1 Analysis vs. monitoring

One application of streaming data visualization is *process monitoring*, common in e.g. the Norwegian oil and gas industry. Process monitoring has existed as a discipline since the early years of the Industrial Revolution, and has only recently been computerized [57]. Braseth et al. distinguish between *monitoring*, where the operator must pay constant attention to the monitored variables and be ready to intervene at a moment's notice, and *analysis*, where there is more time available to reflect and consider deeper relationships in the data [8]. Monitoring allows the real-time investigation of parameter values, but the investigation of more complex relationships between parameters is prevented as only the current parameter values are displayed. Analysis allows a more detailed investigation of complex relationships between parameters, by allowing plot types that show correlations between different parameters. However, analysis is traditionally performed on a static dataset which is separated from the real-time data stream. The distinction between analysis and monitoring is not without reason, as there is often not enough time available to perform detailed analysis in a real-time scenario. This is illustrated in figure 2.4.

Using animated visualizations to display temporal and streaming data is not a new idea [10, 30]. Process monitoring is given thorough treatment from the perspective of interaction design, e.g. by Braseth et al. [8]. However, we have seen only a small amount of work that explicitly addresses the interactive and modular visual analysis methodology of IVA in a form applicable to real-time processes. Implementing a suitable solution to this problem would result in the *analysis* procedures becoming manageable when the urgency and time pressure is higher. Braseth's distinction between *analysis* and *monitoring* is a key motivation for developing new techniques for visual data analysis in the streaming data, real-time domain.

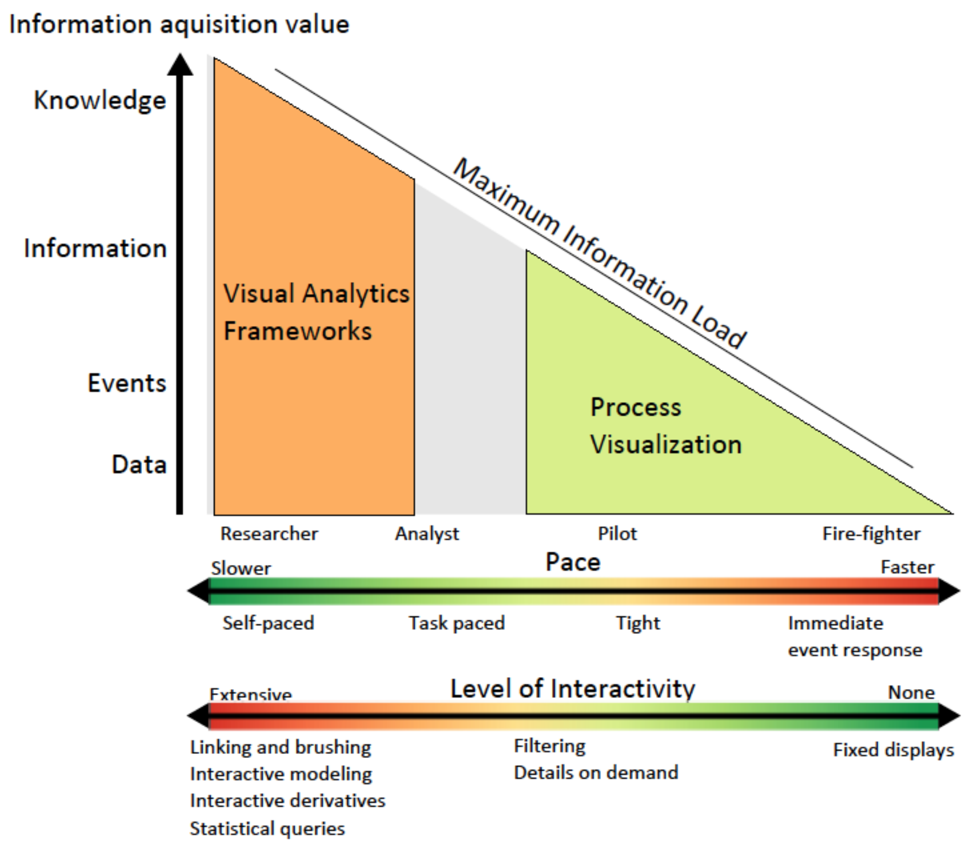


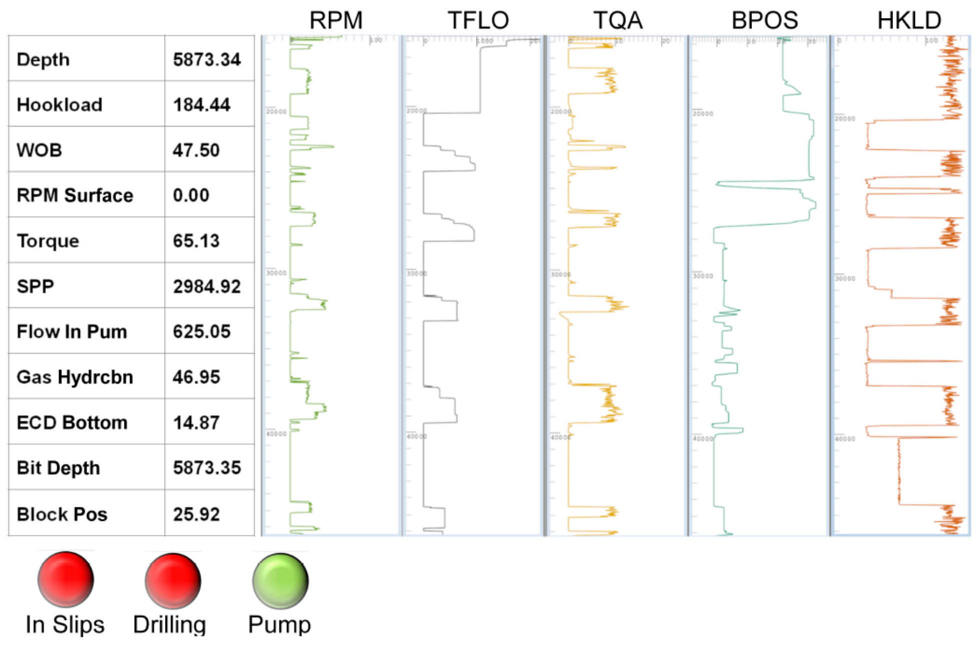
FIGURE 2.4: Information value vs. pace of analysis. As less time becomes available for making a decison, the achievable level of interactivity during data analysis is reduced. We refer to this conceptual range of interactivity as the "analysis-monitoring-continuum". Image courtesy of Ove Daae Lampe [47].

2.3.2 Process monitoring dashboards

Process visualization dashboards (figures 2.5 and 2.6) display the current value of parameters of interest, and often use windowed time graphs in order to provide context to the current sensor values: A number of animated time series are displayed, showing the history of important data values in the recent past[2]. This is called a *dashboard* display [47]. The history is usually provided in order to detect trends, not for performing a detailed analysis of recent data [7].

These streaming data visualization tools are suitable for their domains, but by design provide only a limited capability for the analysis of higher-dimensional relationships in

[2]The amount of history displayed is application-dependent. The key point is that the recent history provides *context* for the current situation, so only the history that is still *relevant* for a skilled operator making continuous decisions, should be displayed.



Depth	5873.34
Hookload	184.44
WOB	47.50
RPM Surface	0.00
Torque	65.13
SPP	2984.92
Flow In Pum	625.05
Gas Hydrcbn	46.95
ECD Bottom	14.87
Bit Depth	5873.35
Block Pos	25.92

FIGURE 2.5: Dashboard for visualization of streaming sensor data from oil drilling. Image courtesy of Ove Daae Lampe [47].

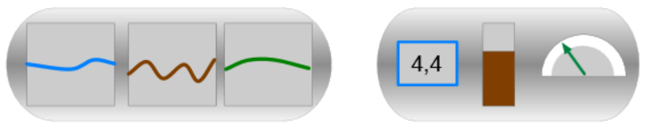


FIGURE 2.6: Real-time process visualization trend lines, with three different display types for the current value. Current values and recent history is easily visible, but deeper relationships between parameters are hidden. Image courtesy of Braseth et al. [7]

the data. There are analysis tasks that have real-time constraints but would still benefit from an ability to perform a more detailed analysis on the fly[3].

2.3.3 Streaming data visualization techniques in InfoViz

Lampe treats some of the challenges of real-time streaming data analysis in his Ph.D thesis [47]. In one example, sensor data from oil drilling operations is investigated[4]. Lampe mentions the challenges of monitoring streaming data in a real-time scenario when only part of the data stream can be stored and viewed at the drilling site. Kernel

[3]A practical example is comparative visualization in a monitoring scenario: If there is a pattern of multiple parameter values (e.g. in temperature, pressure, rotations per minute) that could be used to predict a problem with an industrial process, an operator may want to occasionally compare the current values with the pattern of values that sometimes occurs before a problem. Such analysis is difficult in a monitoring framework, but fits well with IVA.

[4]See Chapter 4, page 33.

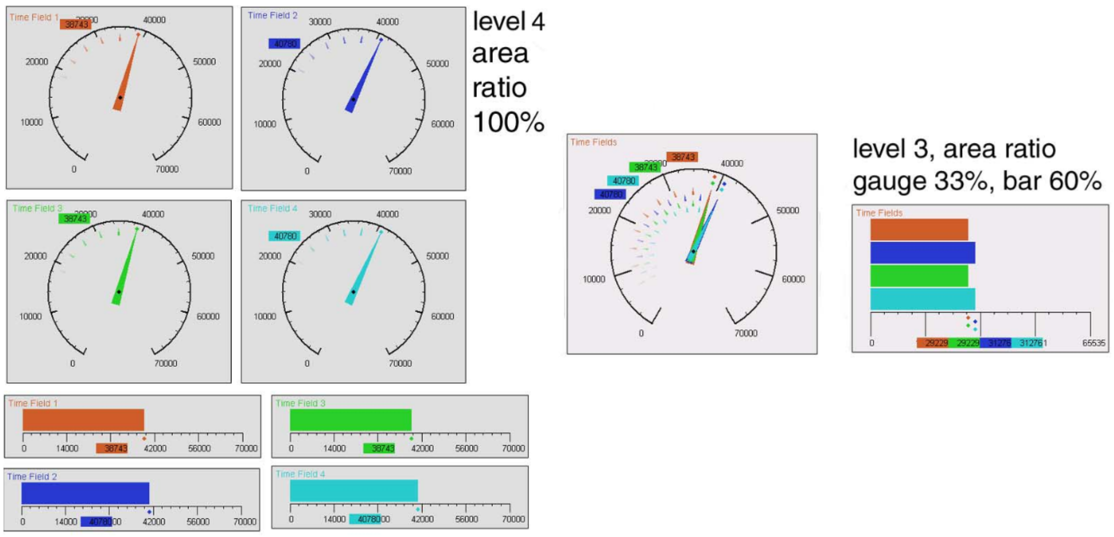


FIGURE 2.7: Levels of detail in process visualization. Instruments display current values and the recent history. The instruments are combined in the right-hand views, reducing monitor space but preserving information content. Image courtesy of Matkovic et al., with minor editorial changes [57].

Density Estimation (KDE), a frequency-based visualization technique, is used to reduce the required storage space and still retain a good view of the information content of the data. As part of the same thesis, Lampe and Hauser describe a novel, GPU-accelerated implementation of KDE [48] which is a lot faster than previous implementations. This implementation is applied to perform two-dimensional KDE on a streaming data source describing real-time ship traffic. The authors note that for this dataset, the KDE calculation would with a previous implementation in fact be prohibitively slow for real-time visualization. This demonstrates that data volumes in streaming data visualization can sometimes be so large that computational performance becomes a problem. The techniques in this paper are demonstrated on simulated streaming data sources[5].

Matković et al. enhanced process visualization by incorporating techniques from InfoViz in the traditional gauge, bar and LED displays [57]. This approach improves the capability of well-known visualization techniques without compromising the available monitor space: The recent history is displayed alongside the current values, and multiple measurements of identical or similar values are robustly combined in a single display. This enables the instant discovery of diverging values. Varying levels of detail support the clarity of the relevant information, as illustrated in figure 2.7. The real-time aspect of these techniques is given by the live data source, and great care is taken to ensure that the displays are accurate and readable in each user scenario. As in most other process

[5]See video by the authors on YouTube: `http://youtu.be/1N5kg_tcG1s`

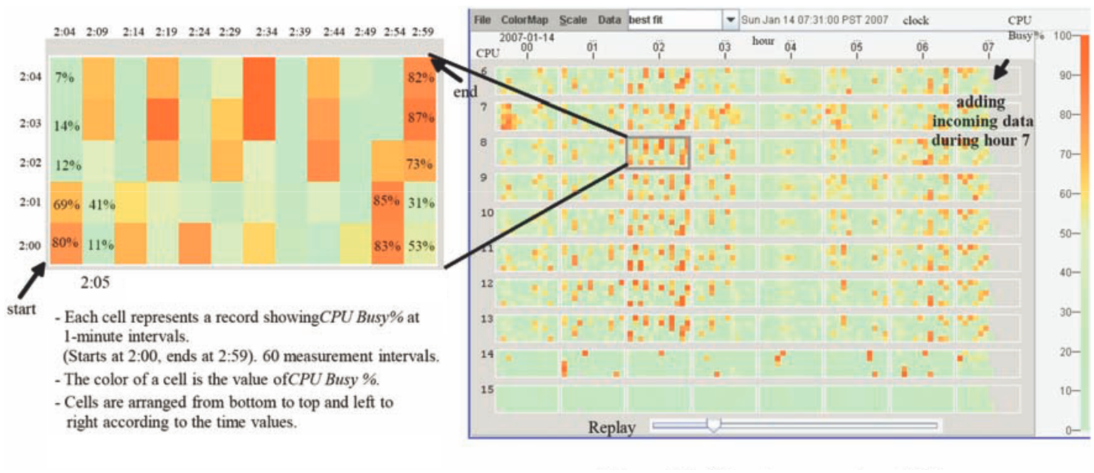


FIGURE 2.8: Density Display for data stream monitoring. Each cell represents average CPU load during one minute, and new data is added as dictated by the layout scheme. There are various strategies for updating the display, with different perceptual tradeoffs. Image courtesy of Hao et al., with minor editorial changes. [24]

visualization displays, immediate attention is emphasized at the cost of the inability to examine higher-dimensional relationships between sensor values.

Hao et al. explored *density displays* for streaming data [24] (figure 2.8). Frequent measurements of CPU load for a large number of servers are continuously inserted in a grid visualization. The authors explore how the large data volumes necessitate frequent updates of this visualization, and how these updates affect operator awareness of earlier data. Three different update schemes are examined, in addition to dynamically changing the resolution of the display as new data is received. Each of these methods have a tradeoff between display consistency, update frequency and potential operator confusion, which exemplifies that streaming data visualizations have more constraints than visualizations of static data.

Krstajić and Keim discussed how traditional InfoViz techniques behave during real-time updates [44]. Seven common visualization types are evaluated with regards to how they change after an incremental update, and how updates affect the viewer's mental model of the data. Scatterplots and line charts change only slightly when a new data point is added, while a basic streamgraph or treemap could get significant layout changes after minor changes to the dataset. This potential *loss of context* in the user's mind is a significant challenge of real-time data visualization. The user's *mental mapping* between the visualization and the actual data will be invalidated if there are large and abrupt changes in the visualization. Suitable transitions must therefore be devised such that

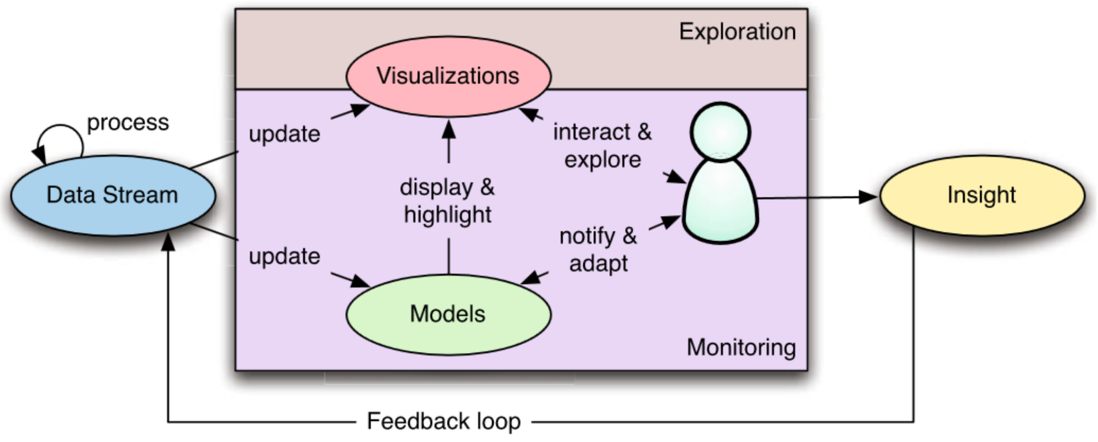


FIGURE 2.9: The dynamic visual analytics pipeline. This interaction loop is introduced by Mansmann et al. [55]. The interaction pattern closely resembles the interaction loop of IVA, which indicates that exploring streaming data IVA might be promising as a generalized approach to realize visual analytics capabilities in a real-time environment. Image courtesy of Mansmann et al. [55]

the visualizations can be updated in a perceptually beneficial way. The paper gives solid treatment of some general concerns of real-time data visualization, and the authors point out that this is an important field where further study would be beneficial.

2.3.4 Systems and network monitoring

Computer network and server administrators constantly operate in a real-time environment. This is a field where streaming data is abundant. McLachlan et al. developed LiveRAC, a visualization system for time series data in data center administration [62]. Hundreds of parameters across thousands of computers (physical and virtual) are monitored, along with event data. This approach uses visualization techniques such as sparklines, line charts, semantic zooming and other focus+context techniques, and focuses on user familiarity and perceptive aspects rather than novel visualization techniques. In this instance, the focus is on *what* to display rather than *how*. The authors have made significant efforts to make simple visualization techniques user-friendly and responsive enough to be useful in an operative environment. The authors concluded the paper with an informal, longitudinal evaluation. Displaying large numbers of charts side-by-side, allowing users to actively manipulate time windows and using linked views, were all techniques that helped the technicians detect errors and inefficiencies that were undetectable through the use of traditional, automated methods.

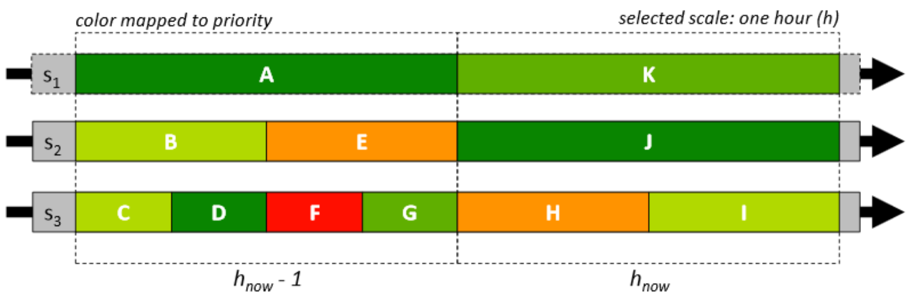


FIGURE 2.10: Relaxed timelines for event visualization. Color is coded to event priority. Older events, on the left, are eventually compressed in space in order to make room for new events on the right. Image courtesy of Fischer et al. [21]

Mansmann, Fischer and Keim coined the term "Dynamic Visual Analytics" (figure 2.9) as "the process of integrating knowledge discovery and interactive visual interfaces to facilitate data stream analysis and provide situational awareness in real-time" [55]. The authors point out that dynamic data yields additional challenges to those of static data sets, both for automated and visual analysis techniques: The data volumes might be unpredictable, the user could experience information overload, the data might behave in unexpected ways and the extreme values of parameters might be unknown. It is also noted that different visualization techniques have varying suitability towards incremental change where a new data point is added, as later explored in detail by Krstajić and Keim [44] (see above). Little research has focused on the applicability of Visual Analytics to real-time analysis, and this topic is assumed to be a rewarding avenue for further research [55].

Fischer et al. developed the Event Visualizer application, which is designed to facilitate real-time analysis of events in a system administrator context [21]. This application uses the concept of *relaxed timelines* (figure 2.10), where system events are color-coded and mapped to rectangles on a horizontal timeline. The timelines are embedded in a dynamic interface which facilitates panning and zooming. The authors discuss automated scoring algorithms which affect event priority, and information drill-down is facilitated through map integration and detail views. The authors' case study clearly illustrates that good visualization tools are required for a timely, operative response during scenarios of up to 420,000 events per hour.

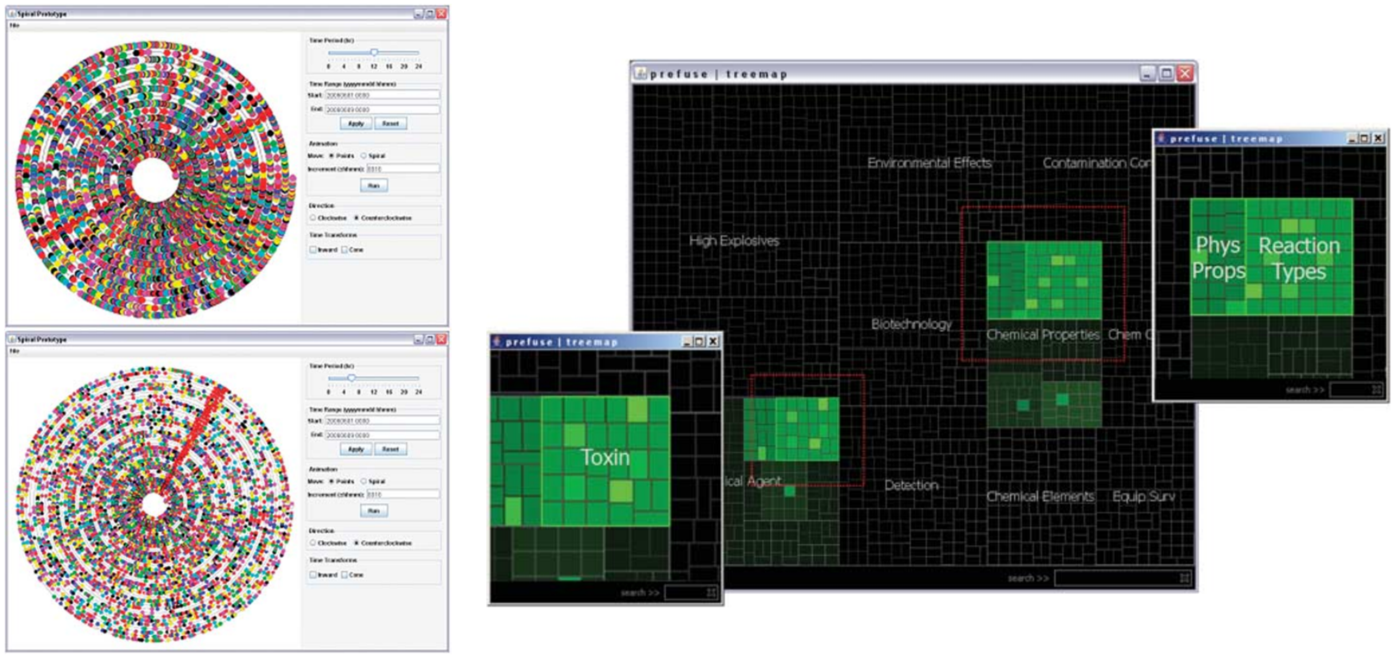


Figure 2.11: On the left: Dynamic Spiral Timelines. Time is mapped to a revolution on the spiral, with the top view having a period of 12 hours and the bottom 6 hours. These spirals display a database of cell phone calls. On the right: Dynamic Tree Map of web searches for specified keywords. Higher luminosity means that this term was searched for more recently. Images courtesy of Chin et al., with minor editorial changes. [11]

2.3.5 Textual analysis

The Internet, the news media and other outlets of human-readable information daily produce massive amounts of textual data. Chin et al. made a comprehensive review of this subject [11]. The authors make adaptions to well-known visualization techniques in order to monitor data streams in real time: Spiral timelines, dynamic tree maps, dynamic tree views, graph views and multi-resolution networks are explored and evaluated. The first two of these are displayed in figure 2.11. For most of the visualizations, *color* is the cue selected for event age. Applications are real-time monitoring of cell phone calls and web searches, presumably for the purpose of the United States' national security. The authors perform an evaluation of the strengths and weaknesses of each technique, and then perform a usability study where each technique is used to complete a specified task. The techniques show significant results of enabling identification of relevant information in real time.

Krstajić et al. introduced visual techniques for analyzing patterns in a massive stream of online news articles [46]. This work aggregates data from a database of thousands of

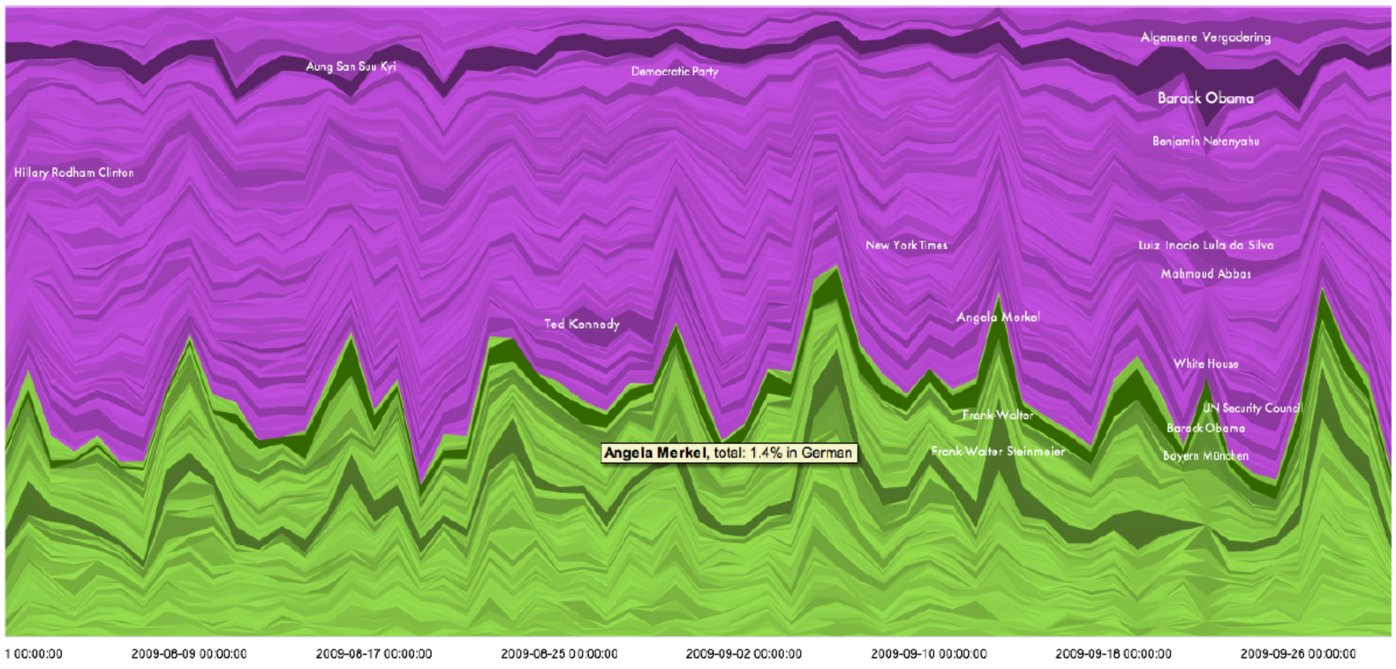


FIGURE 2.12: Visualization of aggregated keyword data for massive news databases. The visualization is updated in real time. Image courtesy of Krstajić et al. [46]

news sources, updated in real time. News articles are automatically decomposed into the concepts they discuss, before these concepts are visualized by relative count and time (figure 2.12). Inspiration has been drawn from the ThemeRiver visualization [29]. A different work introduces sentiment analysis on the same data source, where the sentiment of news articles is correlated with both time and space in real time [45]. These papers demonstrate the impressive aggregation of very large text databases, but the streaming/real-time and urgency aspects are not treated in detail.

2.4 Visualization and analysis of time-oriented data

While InfoViz techniques for *streaming* data still represent a relatively new and unpaved area, the visualization and analysis of *time-oriented* data has received a lot of attention in the literature. The time series, most commonly represented with a line chart, is a very common representative of such data. However, many other possibilities exist. Aigner et al. present a very comprehensive review of the different types of time-oriented data, along with visualizations that are used to represent them [1, 2]. Spiral graphs [79], ThemeRiver [29], and temporal clustering [76] are all examples of visualizations that can be successfully applied to time-oriented data. Which visualization is most suitable, depends on the analyst objective and the characteristics of the data in question. In the examples we have mentioned here, spiral graphs work well when temporal cyclicity is an important feature of interest, ThemeRiver works well for exploring categorical data or large time-stamped text collections, while clustering is a good technique to display *aggregated* data, temporal or otherwise [2]. A comprehensive review of time-oriented data and visualization is beyond the scope of this thesis, but we note that this is a subject which has been studied in exquisite detail.

We now discuss some different approaches and concerns that are relevant to the subject of visualizing time-oriented data.

2.4.1 Symbolic representation of time series

Lin et al. developed SAX [51], a symbolic representation of time series which provides a high level of abstraction while facilitating motif enumeration and anomaly detection. The methodologies surrounding SAX are summarized and implemented in the VizTree application (figure 2.13), facilitating rapid classification, search and anomaly detection in large time series. This technique transforms the time series into a normalized Piecewise Aggregate Approximation (PAA), which is similar to the approximation by a linear combination of box basis functions. The PAA is then further discretized into an alphanumeric string, where each letter in the alphabet denotes a particular range in the PAA and each letter in the string represents a fixed time span [51]. Notably, the abstraction algorithm in SAX is local. This is a subtle but important point which makes it suitable for use on streaming data of unbounded size. The SAX representation allows the data to be encoded in a modified suffix tree, which is a powerful visualization technique for motif discovery [52]. This allows allows visual or automated classification of recurring patterns in time series, which is useful from the perspective of knowledge discovery [50][6].

A suggested application of these techniques is the analysis of time series telemetry data in the final hours before a space rocket launch, supporting the binary decision of whether or not to launch the rocket [53]. This scenario exhibits a decision pressure which is very much in the real-time domain. Although the authors don't explicitly discuss the challenges of streaming data monitoring, the scenario clearly demonstrates a need for

[6]As a historical note, it appears that the application of motif discovery through a novel symbolic representation [50] was explored before the details of the symbolic representation (SAX) was fully articulated in the literature [51].

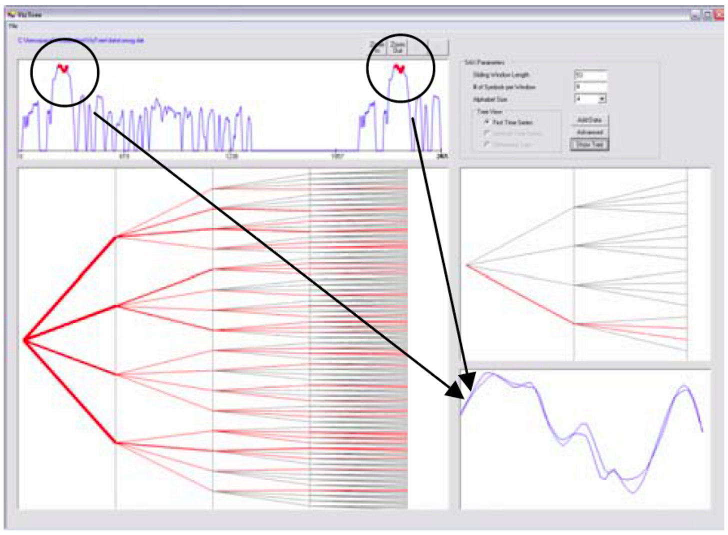


FIGURE 2.13: The VizTree application

A screenshot of the VizTree application, here used to detect an anomaly in an ECG dataset. Image courtesy of Lin et al. [53]

data analysis methodologies that provide decision support in real-time situations. SAX is a successful technique which is cited in many other papers, as outlined in a review by Fu [22].

2.4.2 Navigating in massive time series

When working with very large time series, it can be difficult to find the required part of the time series due to limited monitor space and human perceptual abilities. SAX is a symbolic aggregation technique, which greatly reduces the information content in order to detect interesting features. When navigating in the details is required, we must use a different approach. Kincaid has developed SignalLens (figure 2.14), which imitates a magnifying glass in order to display the focus+context when investigating signal traces from electronics [41]. These time series are from 10^4 to 10^7 points in size. Kincaid's lens implements a lightweight optical simulation, and also includes some pattern recognition capabilities to allow automated search.

An alternative to the *space distortion technique* used by Kincaid, is to reduce the vertical space of the line chart, stretch it over multiple lines and instead use color to emphasize signal magnitude (figure 2.15). These techniques are known as *pseudo coloring techniques*, and a prominent example is *two-tone pseudo coloring* (2TPC) [69]. 2TPC uses a skewed color map to draw each vertical portion of the time series, and yields a very

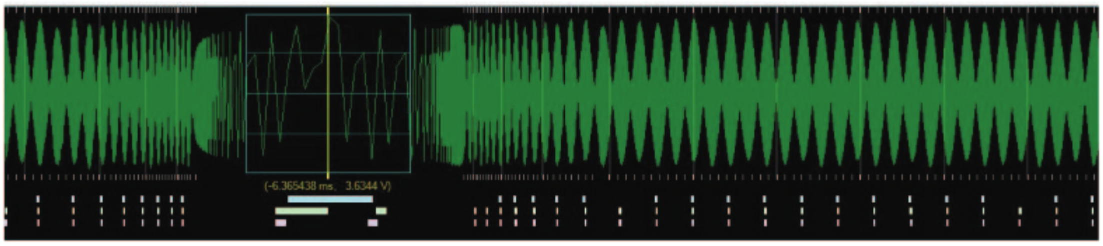


FIGURE 2.14: Kincaid's SignalLens, showing a space-distorted focus+context view of a large amplitude-modulated signal trace. Boxed region is of equidistant samples, while the the area on its sides gradually decreases the magnification level until only the unmagnified signal is displayed. Image courtesy of Robert Kincaid [41].

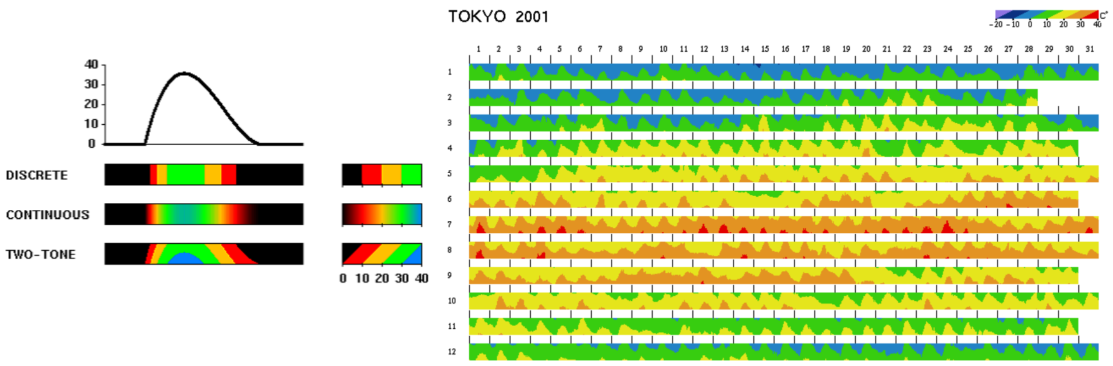


FIGURE 2.15: Two-tone pseudo coloring. Left-hand diagram shows how values are mapped to a color scheme. Right-hand side uses this technique to display the hourly temperature variation in Tokyo throughout a whole year. Image courtesy of Saito et al. [69]

intuitive plot where both color and shape cues are provided. This makes it easy to see minima, maxima and patterns. 2TPC is very intuitive and can display large amounts of data in a small space, provided the data and format of the visualization is appropriate. The *horizon graph* is a very similar technique, which is suitable for time series which have a natural center value [20, 66]. Jerding and Stasko's *information mural* technique is a different method, which can also be used for drawing large time series in a limited space [34]. The information mural uses coloring and anti-aliasing to ensure that *over-plotting* does not greatly reduce the information content of visualizations, when there is limited space available for drawing a large amount of data. The authors use this technique to draw an expressive time series plot of daily sun spot observations for a 143-year period, using only the width of an A4 page.

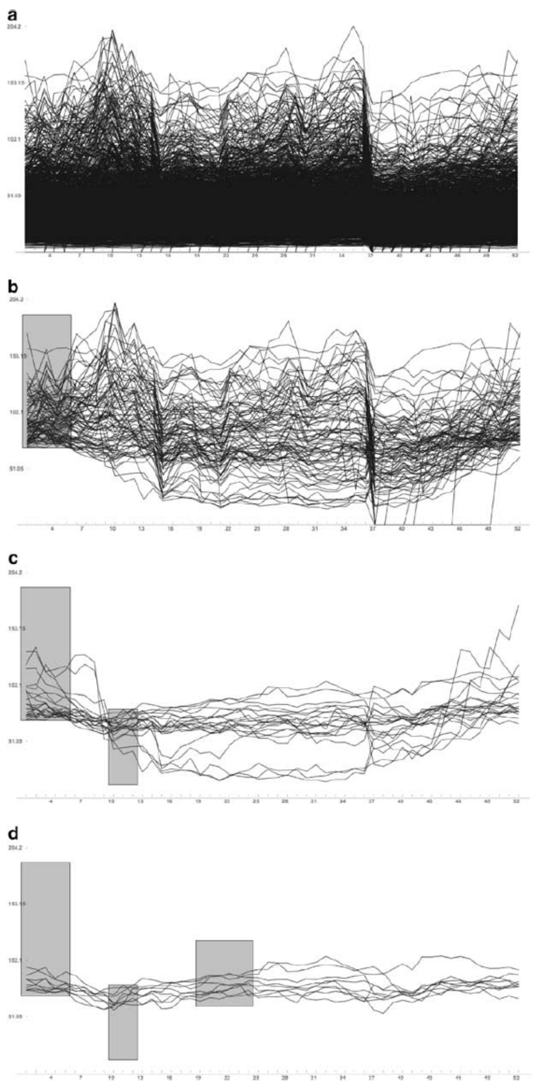


Figure 2.16: Timebox widgets, used to successively narrow down a selection of time series rom a large initial query. Image courtesy of Hochheiser et al. [32].

2.4.3 Searching in time series

Although visual techniques allow us to include human insight and pattern recognition in the investigation, automated tools are sometimes required to find a particular feature in a large time series. This subject has been given a lot of attention.[7] Regarding visual techniques in particular, Wattenberg created QuerySketch which lets the user sketch a portion of a time series and automatically detect similar patterns [77]. Earlier techniques for time series search required manual specification of the requested data characteristics. Wattenberg's approach creates a layer of abstraction over these methods, exemplified by the *Euclidean distance* metric. This allows the user to use free-form sketch queries,

[7]A detailed review of such techniques is beyond the scope of this thesis, but an article by Lin et al. has a comprehensive review of query-by-content algorithms for large time series [50].

which are more intuitive due to the visual time series representation that is used in practice.

Hochheiser, Shneiderman et al. introduced a different technique to specify queries in large time series databases: *Timeboxes* (figure 2.16) specify the requested ranges in both time and value, and can be combined in order to narrow down a query. This approach is combined with the notion of *envelopes*, which give a rough visual indication of the distribution of the time series in question [31].

While both these techniques work on time series of arbitrary size, the latter is adapted to the situation where we are searching among a large number of *overlapping* time series. An example would be hundreds of simulated temperature curves in a climate model. Timeboxes and envelopes are implemented in the *TimeSearcher* application [31], which provides an interface for applying this technique to large databases of time series. The authors also demonstrate the suitability of this technique for querying a genome database, an example of data which is not time-dependent [32].

2.5 Summary

This concludes our discussion of the scientific context for this thesis. There are many good and interesting works on topics that are closely related to our main area of focus, and we conclude that streaming data visualization is a subject which appears worthwhile to investigate closer. In the next chapter, we discuss the conceptual and practical challenges of analyzing streaming data in real time, in the context of Interactive Visual Analysis.

Chapter 3

Interactive Visual Analysis methods for Streaming Data

This chapter focuses on the challenges that appear when adapting Interactive Visual Analysis to its use on streaming data in real time. We explore the practical problems and conceptual background for these challenges, and suggest solutions that are compatible with the interactive and dynamic nature of IVA. We explore areas where the extension of IVA to streaming data introduces additional complexity, and suggest techniques that allow users to manage this complexity. While many of these concerns are unique to IVA, some of them are relevant for visualization of streaming data in general. Some of our techniques are exemplified on video throughout this chapter, before they are subjected to evaluation and demonstration on real data in Chapter 4.

3.1 Motivation

As we have seen, there has been relatively little work focusing on visual analysis techniques for real-time and streaming data [55]. One reason is that treating the data as streaming is more complex than treating it as static. Another reason is that such real-time analysis capability only covers a minority of the cases where visual analysis is useful. Static analysis techniques like traditional IVA can be very useful even on datasets that are dynamic. An analyst could for example copy a slice of the live data into a suitable format such as CSV and then perform IVA on it. This approach simplifies the analysis by treating the data as if it was static. Depending on the task at hand, this could be an adequate technique to get analysis results in a timely manner. We use the term *decision pressure* to denote the urgency of making sense of new data as it arrives, and then making decisions based on the observations. The requirement to make decisions from observations is central to the analysis of streaming data: If our hunt for knowledge and insight is not followed by the requirement for timely practical action, it is clearly possible to use a more leisurely approach for data analysis even if the data rate is high (figure 3.1).

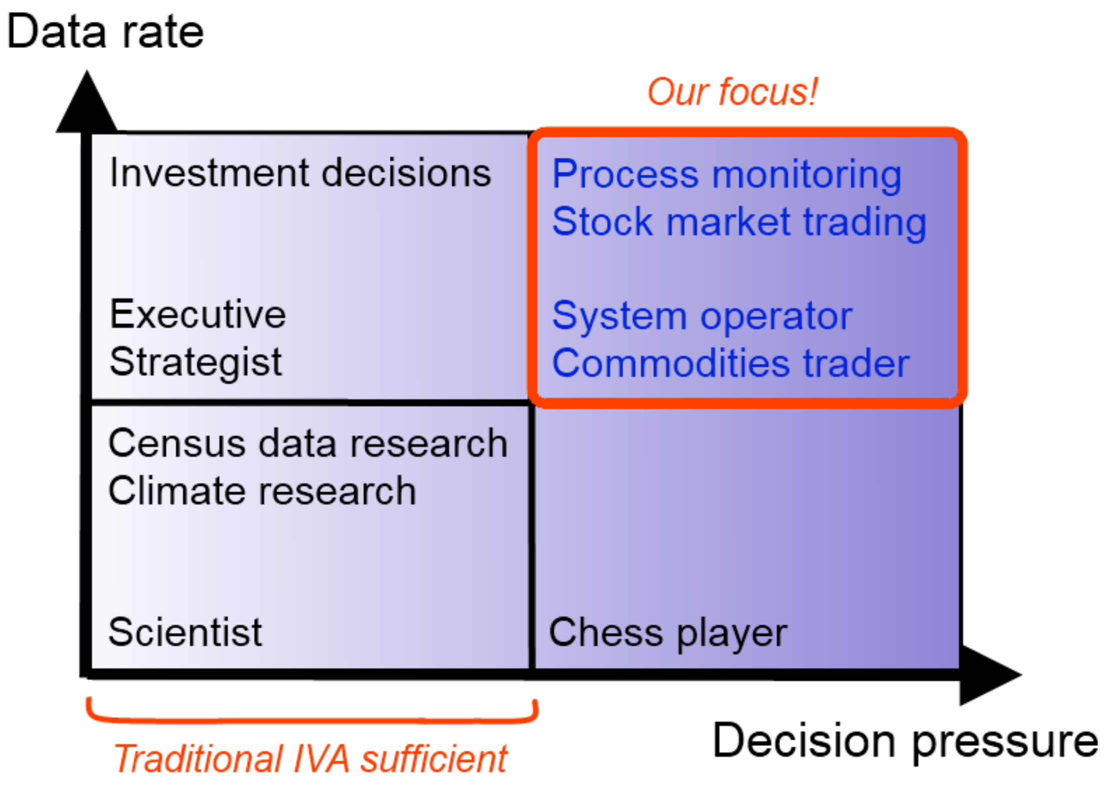


FIGURE 3.1: Data rate vs. decision pressure. Analysis techniques differ, depending on the data rate and the urgency and rate at which decisions must be made.

The key concern for whether treating the data as static is appropriate or not, is whether the insights from the analysis process are strongly affected by real-time fluctuations in the data – in other words, whether "surprises" in the data stream are likely to occur after the analyst has exported the data for analysis, but before the analysis is complete[1]. If this is the case, conclusions drawn from the static analysis could be invalidated by the time the analysis is finished, resulting in incorrect decisions.

[1]From the perspective of a wind farm operator, examples of this could be sudden changes in wind conditions, sudden changes in temperature that affect the safety of maintenance crews etc.

The discussion of *decision pressure* above implies that the *rate* at which decisions are made, is also important. If making practical decisions from observations in the data is a rare occurrence that can be done at the operator's discretion (for example if it is very expensive to perform actions that modify the monitored process), the decision pressure is correspondingly low. In this case, it might be appropriate to use the traditional method of treating the data as static, giving less attention to the real-time aspect of the scenario. This means that decisions become strategic rather than tactical, moving us away from the real-time domain as indicated on figure 2.4. The *purpose* of the analysis is clearly also important. Data rate does not necessarily correlate with the decision pressure, so a high data rate is not in itself a reason for using more dynamic visualization techniques[2].

However, dynamic visualization methodologies that incorporate the real-time aspect are still applicable when the decision pressure is lower. The only requirement for using streaming data visualization techniques, is that the data source updates our data in real time.

3.2 Initial assumptions and challenges

Many challenges show up in visual data analysis when considering streaming rather than static data. Some of these are technical in nature, while others are caused by the fight between the finite human attention and the inherent, inexorable progress of time. We will assume that the analysis framework in question receives a stream of discrete data in a spreadsheet-like format, one row at a time. This is already a simplification over the more general form where each dimension could have a separate data rate, but it is a simplification we consider reasonable for the purposes of exploring streaming data IVA.

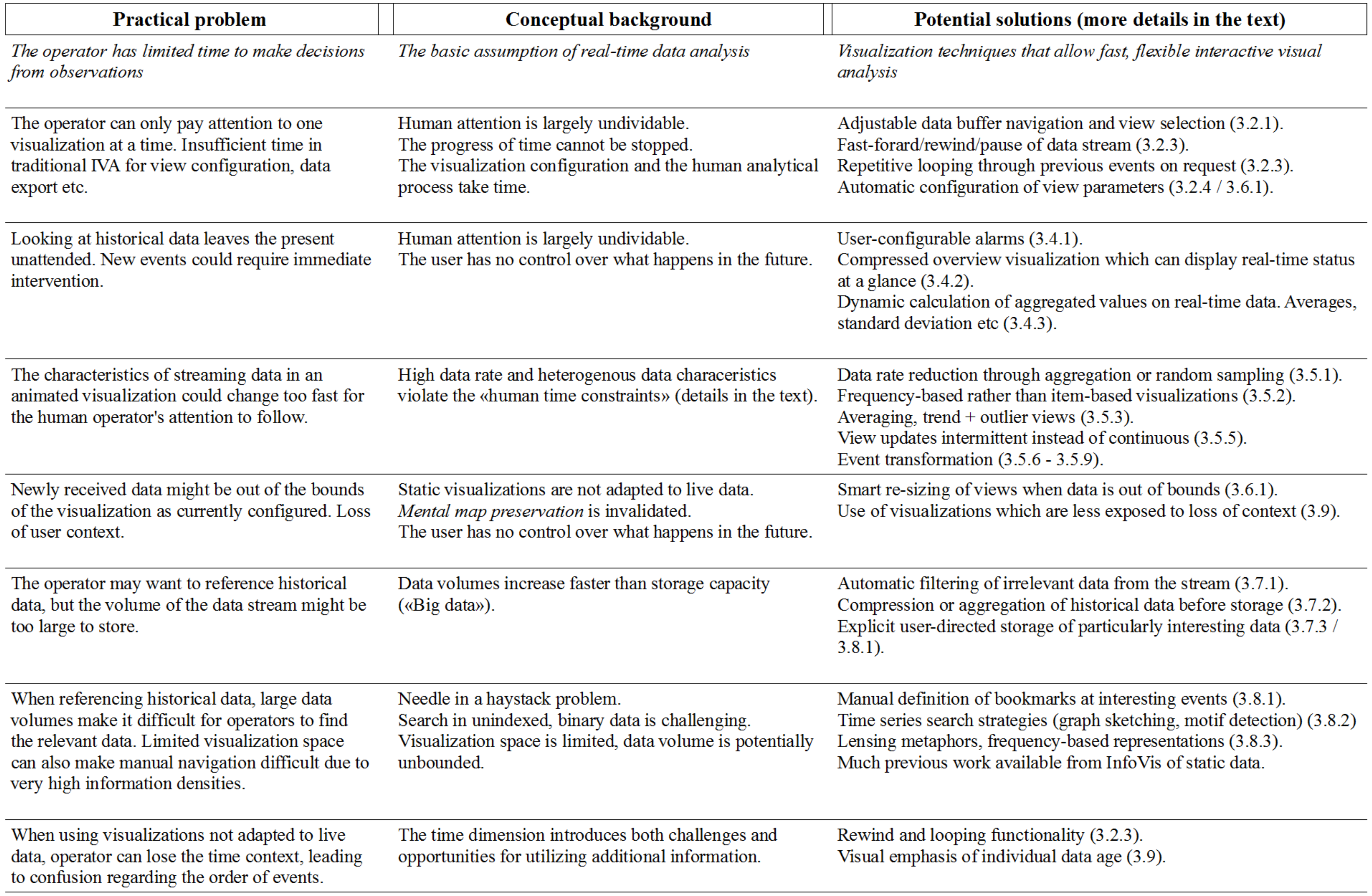
[2]As an example, social networks gather huge amounts of data on the behavior of millions of users, on a minute-to-minute basis. The streaming aspect of this data is irrelevant for a researcher using this data to investigate e.g. the development of smoking trends among teenagers.

Manual pre-processing of data is impossible in a scenario where the data is streaming. Pre-processing is usually an important part of the Visual Analytics process [40], and may involve the removal of corrupted data, interpolating or filling in missing data points, removing outlier data points or any other operation which cleans up the source data. This process is very dependent on the data in question, and is often manual. Real-time analysis necessitates the automation of this process, since we neither have the whole dataset nor have the time to manually inspect the data prior to analysis. This obviously necessitates a lot of care in the implementation of the data source and the visualization framework, since unexpected discrepancies in the data can in the worst case lead the analyst to draw incorrect conclusions.

When investigating IVA of streaming data, we mostly use simple visualization techniques: Scatterplots, histograms and line charts. As noted by Krstajić et al. [44], different visualization techniques show great variation in the degree to which they can be adapted to real-time, continuous updates as new data is received. Loss of context and user confusion are common when using advanced visualization techniques on streaming data. This illustrates that real-time data visualization is a broad field which requires consideration of both the technological and human aspects in order to provide useful tools.

Hence, a recurrent theme throughout this chapter is the unique challenges an operator faces when using an imagined software tool to create and make sense of visualizations of streaming data. We imagine a person monitoring a live feed of data subject to the constraints defined above, with the responsibility to make timely decisions or provide advice based on the characteristics of the data stream. For each of the practical challenges this person faces, there is a corresponding conceptual background for the problem and multiple ways we could attempt to resolve the challenge.

We now provide a high-level overview (figure 3.2) of the challenges that show up when performing IVA on streaming data in real time. The challenges are described by how they manifest themselves as a practical problem, the conceptual background for this problem and a brief description of our suggested solution. This should make it easier to follow the detailed reasoning and examples below, which roughly follow the structure of the overview.



Practical problem	Conceptual background	Potential solutions (more details in the text)
The operator has limited time to make decisions from observations	*The basic assumption of real-time data analysis*	*Visualization techniques that allow fast, flexible interactive visual analysis*
The operator can only pay attention to one visualization at a time. Insufficient time in traditional IVA for view configuration, data export etc.	Human attention is largely undividable. The progress of time cannot be stopped. The visualization configuration and the human analytical process take time.	Adjustable data buffer navigation and view selection (3.2.1). Fast-forard/rewind/pause of data stream (3.2.3). Repetitive looping through previous events on request (3.2.3). Automatic configuration of view parameters (3.2.4 / 3.6.1).
Looking at historical data leaves the present unattended. New events could require immediate intervention.	Human attention is largely undividable. The user has no control over what happens in the future.	User-configurable alarms (3.4.1). Compressed overview visualization which can display real-time status at a glance (3.4.2). Dynamic calculation of aggregated values on real-time data. Averages, standard deviation etc (3.4.3).
The characteristics of streaming data in an animated visualization could change too fast for the human operator's attention to follow.	High data rate and heterogenous data characeristics violate the «human time constraints» (details in the text).	Data rate reduction through aggregation or random sampling (3.5.1). Frequency-based rather than item-based visualizations (3.5.2). Averaging, trend + outlier views (3.5.3). View updates intermittent instead of continuous (3.5.5). Event transformation (3.5.6 - 3.5.9).
Newly received data might be out of the bounds of the visualization as currently configured. Loss of user context.	Static visualizations are not adapted to live data. *Mental map preservation* is invalidated. The user has no control over what happens in the future.	Smart re-sizing of views when data is out of bounds (3.6.1). Use of visualizations which are less exposed to loss of context (3.9).
The operator may want to reference historical data, but the volume of the data stream might be too large to store.	Data volumes increase faster than storage capacity («Big data»).	Automatic filtering of irrelevant data from the stream (3.7.1). Compression or aggregation of historical data before storage (3.7.2). Explicit user-directed storage of particularly interesting data (3.7.3 / 3.8.1).
When referencing historical data, large data volumes make it difficult for operators to find the relevant data. Limited visualization space can also make manual navigation difficult due to very high information densities.	Needle in a haystack problem. Search in unindexed, binary data is challenging. Visualization space is limited, data volume is potentially unbounded.	Manual definition of bookmarks at interesting events (3.8.1). Time series search strategies (graph sketching, motif detection) (3.8.2) Lensing metaphors, frequency-based representations (3.8.3). Much previous work available from InfoVis of static data.
When using visualizations not adapted to live data, operator can lose the time context, leading to confusion regarding the order of events.	The time dimension introduces both challenges and opportunities for utilizing additional information.	Rewind and looping functionality (3.2.3). Visual emphasis of individual data age (3.9).

FIGURE 3.2: An overview of the challenges, concepts and solutions covered in this chapter.

Many of the challenges mentioned above are applicable to streaming data visualization in general, and not only to Interactive Visual Analysis. Since IVA is an interactive and iterative technique, we also need to discuss how Focus + Context in the form of data brushing on dynamic linked views adapts to real-time situations. As we discuss later, the real-time aspect increases the complexity of linking and brushing, as well as that of other established IVA techniques. The pressing real time aspect provides many additional challenges for data visualization.

3.2.1 Terminology and basic visualization framework

We start by introducing a rudimentary IVA framework: Four linked views, each of which can be configured to display a scatterplot, a histogram or a line graph (see figure 3.3). Data items highlighted in one of the views are also highlighted in all the other views. Inspired by ComVis [58], this layout constitutes a lightweight environment in which the "Show & Brush loop" can be performed. The three available view types have the advantage that they preserve user context comparably well when incrementally updated [44], and therefore provide a good basis for the following discussion.

We imagine that this interface is connected to a data source, where data items arrive one at a time, with a fixed time interval between each data item. Assuming finite storage

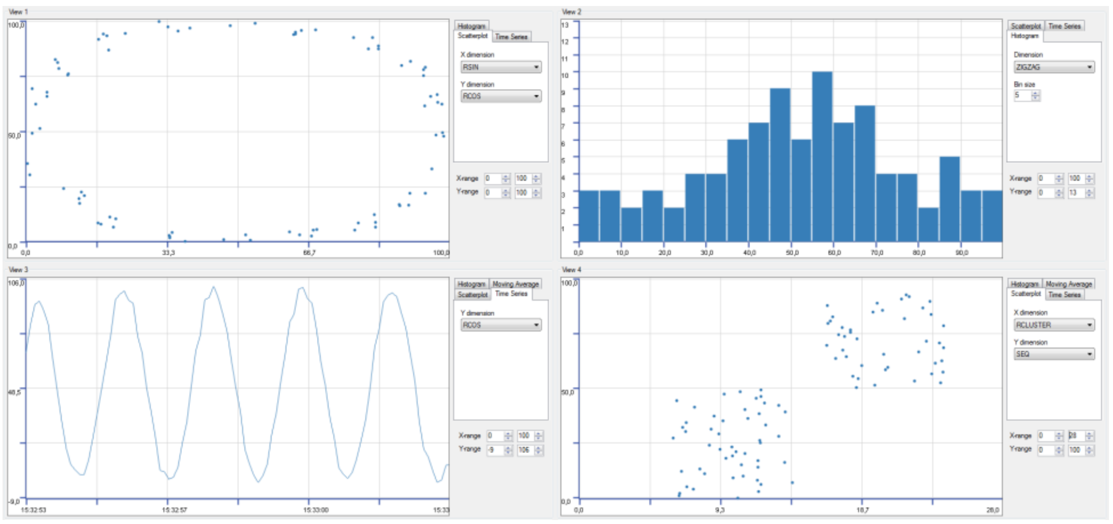
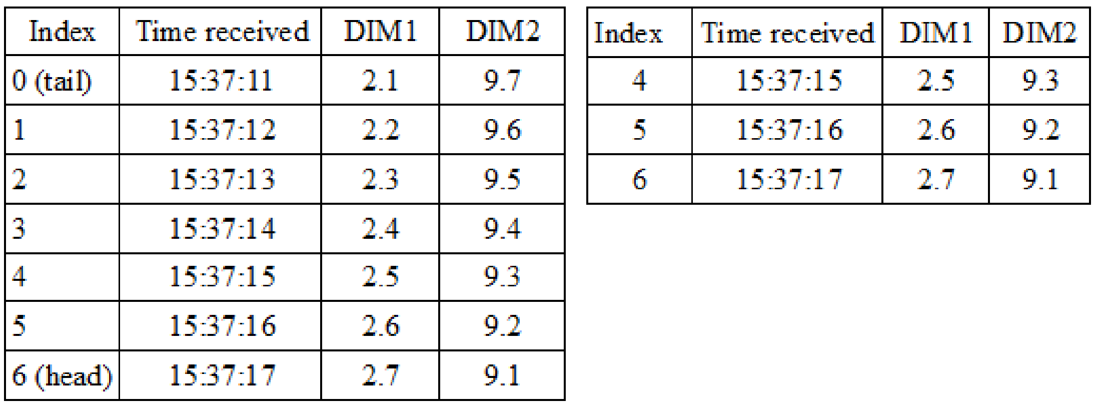


FIGURE 3.3: Simple IVA framework: Four linked views, where the view type can be selected with the tabs, parameters selected with dropdown boxes and the view range is selected with the numeric selectors in the bottom-right corner of each view. See Appendix A for annotated high-resolution screenshots. The video *Demo scenario 1* demonstrates the whole interface in action, and it is introduced in Chapter 4.



Index	Time received	DIM1	DIM2
0 (tail)	15:37:11	2.1	9.7
1	15:37:12	2.2	9.6
2	15:37:13	2.3	9.5
3	15:37:14	2.4	9.4
4	15:37:15	2.5	9.3
5	15:37:16	2.6	9.2
6 (head)	15:37:17	2.7	9.1

(a) Data buffer, length 7

Index	Time received	DIM1	DIM2
4	15:37:15	2.5	9.3
5	15:37:16	2.6	9.2
6	15:37:17	2.7	9.1

(b) View window. Length 3, start index 4

FIGURE 3.4: Illustration of the used data structures.

capabilities, these items must be stored in a finite buffer before drawing them on screen. We choose a FIFO queue of fixed length, which will be referred to as the ”data buffer”. Data items leaving the buffer are permanently dropped.

What part of this buffer should be drawn in the views? The data buffer can be large in both size and time span, and the user focus and the computational resources are limited. Therefore, we argue that a moving window of user-defined size is the correct selection to draw in the views. This moving window will initially encompass the data items which were received most recently, and by default sticks to the head of the data buffer. We refer to this as the ”view window” (see figure 3.4). The metaphor of a moving window of focus allows the flexible selection of user context, and the situation where an operator wants to display all available data points emerges as a special case where the view window size equals the data buffer size.

The traditional process visualization dashboard where the recent values of some important sensor values are drawn as line graphs, occurs as a special case of such a layout.

3.2.2 The conflict of attention

A fundamental challenge when analyzing streaming data, is how to relate to time itself. A user can only pay close attention to one visualization at a time. Time will keep progressing no matter what the user chooses to do, and this implies that there is a fundamental conflict about what to do with the available time. We use the terms ”focus time” and ”data time” to distinguish between what the operator sees and the data that

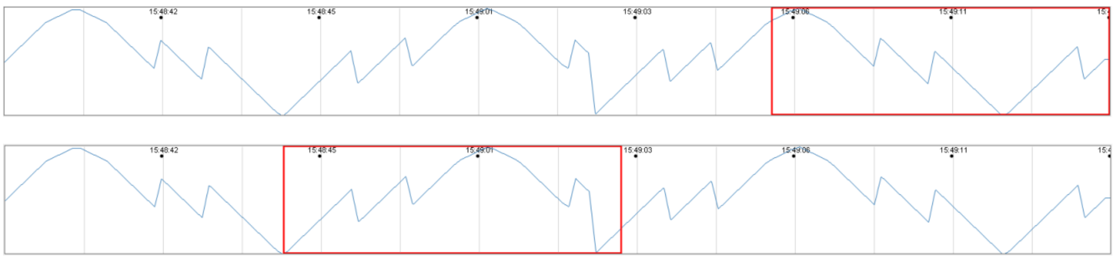


FIGURE 3.5: Buffer overview with the view window highlighted in red, before and after shifting the view window to a previous region of focus. In the top example, the view window encompasses the region at the head of the FIFO queue, selecting the most current data. In the second example, the view window has been moved towards the tail of the queue, so that it selects older data items.

currently arrives through the data stream. In a normal dashboard display, focus time and data time will be identical as long as the operator sits at the console and monitors the data stream. If something happens in the data stream, it will be immediately obvious.

However, we can imagine scenarios in which the user would like to *stop* the dynamic visualization in order to closer inspect something that may have occurred in the data stream. In such a scenario, focus time has been stopped and will inexorably diverge from data time. Events which now occur in data time (i.e. what is *actually* happening) will not be obvious to the operator, and depending on the data characteristics, the discrepancy between the focus time and data time will grow as time passes. Pressing real-time events could occur while the operator's attention is elsewhere. User time and attention are scarce resources which must be invested with care.

3.2.3 Time navigation

Our basic visualization framework implies that the operator will by default only see the latest data items that arrive from the stream. To provide context for the view window, one obvious solution is to also display a line graph of a prominent dataset dimension of the entire data buffer, with the contents of the view window highlighted. A line graph maps the the time dimension directly to the X axis, which provides an anchor for the operator's mental model of the progress of the process behind the data stream.

This intuitively yields an interface for manually selecting which portion of the data buffer to display in the view window: Dragging the view window across the data buffer moves the view window's region of focus (see figure 3.5).

Data playback

By default, the view window will stick to the head of the data buffer, with new data items constantly being inserted ”on the right” and old data items leaving ”on the left”. But if an interesting event occurs in the data stream, the operator might want to freeze the visualization in order to investigate this event and its context more closely. Such an investigation could consist of bringing up new views, looking at other dimensions or using the Show & Brush technique to investigate higher-dimensional properties of the data.

To facilitate this type of analysis, we introduce a playback metaphor: The operator can choose to pause the visualization, which causes the view window to stick to the data items currently in view. Meanwhile, the data buffer keeps receiving new data items from the stream and discards the oldest, so the view window will gradually shift towards the tail of the data buffer. The end result is that the operator can now investigate the features of the relevant data items, without paying attention to the head of the continuously updating data buffer. (This also implies that focus time and data time diverge).

Our playback metaphor mimics the terminology from an old-fashioned cassette deck: Play, pause, fast-forward and rewind (interface illustrated in figure 3.6):

- Play makes the view window follow the data buffer, perhaps shifted back in time from the head of the buffer. If this is the case, the animated views will display the development of the data stream in the recent past. Otherwise, they will display the current data.
- Pause freezes the view window, as described above.
- Rewind gradually shifts the view window from the present into the past (towards the tail of the data buffer), animating the views during the process.
- Fast-forward gradually shifts the view window towards the head of the data buffer.

Playback actions only affect the position of the view window with respect to the data buffer. The data buffer is always updated independently.

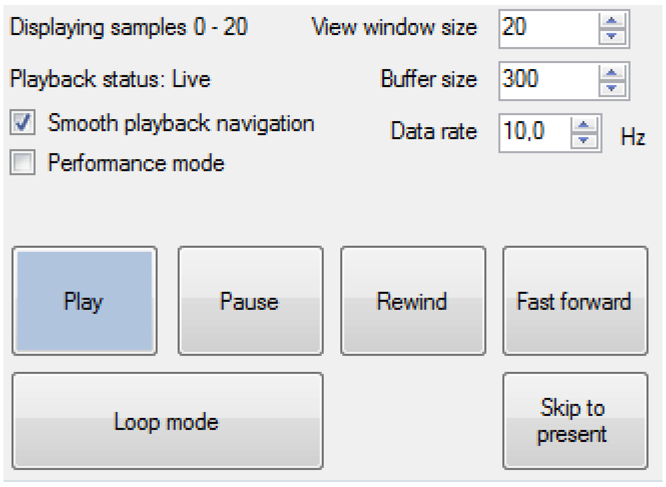

Displaying samples 0 - 20
Playback status: Live
[✓] Smooth playback navigation
[ ] Performance mode
View window size: 20
Buffer size: 300
Data rate: 10,0 Hz

FIGURE 3.6: Playback and buffer control panel

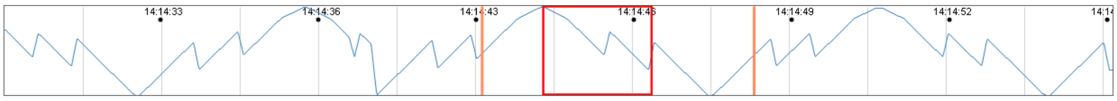
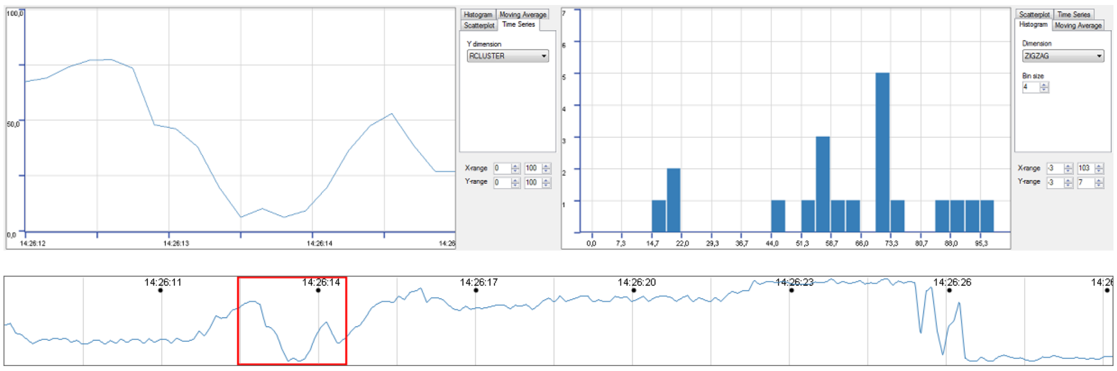


FIGURE 3.7: Illustration of the loop mode. The region between the orange delimiters will be repeatedly looped until this region leaves the data buffer, at which time regular playback will continue. A video of this can be seen in *Demo scenario 2*, which is treated in detail in Chapter 4.

Looping

We also need to consider the case where an interesting event happened over a time longer than the length of the view window, or where we otherwise want to replay and investigate the event multiple times. We introduce a looping function, where the operator is allowed to select a start and end point in the data buffer (see figure 3.7). In loop mode, the view window will move at the same pace as the data buffer between these two points, "replaying" the event for the operator as many times as necessary. Optionally, the looping process can be accelerated by fast-forwarding at the same time.

We have now seen one dimension along which the streaming aspect complicates the IVA process: The analysis can take place on data which is either frozen or live. In the case of live data, the process can happen either on data from the present or from the recent past. This is in contrast to IVA of static data, where there is no inherent time pressure and the dataset is always the same.



Y dimension: RCLUSTER
X-range 0 100
Y-range 0 100

Dimension: ZIGZAG
Bin size: 4
X-range -3 103
Y-range -3 7

FIGURE 3.8: Buffer overview and linked views. During investigation of an earlier event, there are new developments in the present. Fast-forward would allow seeing the development of the histogram until the present, while instantly skipping would give us the details of the current events immediately.

Rejoining focus time and data time

The conflict of attention exists until the analyst has completed his analysis process and decides to return to a real-time view of the data stream. This implies merging focus time and data time again. The physical time between pausing and resuming the data stream is lost forever, so at the best the analyst will have a limited view of what has happened during this interval. We want to minimize the loss of attention during this transition. Using the fast-forward functionality to animate between the data visualized at $\mathbf{T_{stop}}$ and $\mathbf{T_{current}}$ is one approach (see figure 3.8, or the video below). A second option is to skip directly to $\mathbf{T_{current}}$, ignoring everything which has happened during the stop interval. The suitability of each of these methods depends on the visualizations used: Our objective is to ensure that the analyst still has the opportunity to assess whether something interesting happened during the stop. A time-series view which always displays the surrounding context might play well with a simple jump, whereas with a scatterplot we would potentially lose overview of all data points received during the interval. *Change blindness* [72] is also a risk in such scenarios: when switching between two views containing different data, the operator's attention would have to be directly focused on the screen in order not to risk missing important changes. Leaving these choices to operator discretion seems like the most flexible approach.

Video of fast forward: `http://vimeo.com/geirsmestad/iva-fastforward`

Using these interaction tools to navigate between the past and the present, the operator is no longer tied to the monitoring role where only the present is in focus. If time

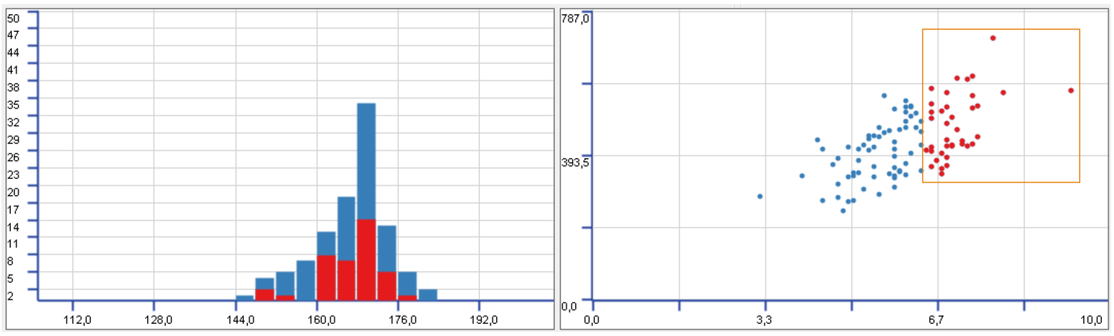


FIGURE 3.9: Brushing operation, investigating correlation between two linked views. The data is from the Hywind offshore wind turbine. Histogram of wind direction to the left, scatterplot of wind speed on X vs. produced electric power on Y to the right. In this example, no obvious correlation is evident in the brushed data.

permits, the operator can move between the *monitoring* and *analysis* roles described in Chapter 2, and hence receive additional capability to what would be possible using only traditional process monitoring techniques.

3.2.4 Automatic configuration of view parameters

Configuration of view parameters, such as view size and dimension, should be as easy as possible in order to allow the operator to direct her time and attention at the data rather than the interface. We will come back to this subject later. For now, we just note that facilitating fast, preferably automated configuration of views, is a very important aspect of interaction design in any applications for streaming data analysis[3].

3.3 Brushing in animated linked views

The brushing metaphor is a simple but powerful IVA technique for higher-dimensional data analysis: Mark a selection of data items in one view, and the same data items will be highlighted in all other views [64] (see figure 3.9). This technique reveals many higher-dimensional relationships in a manner that is intuitive and preattentive.

[3]While the importance of quick, hands-off software configuration is emphasized time and time again by interaction designers, both commercial and research software often violates this by requiring excessive effort to configure. So while this point might seem obvious, it is worth stating again. In software for streaming data analysis, there is simply no option but to conserve the user's time and attention.

In the real-time domain, however, there is an important ambiguity as to what such a brushing operation refers to. We have identified four different scenarios regarding what the user wants to see[4].

1. The data items *in a specified range of the data space* are selected
2. The *specific data items which are visible at the time of brushing* are selected
3. The data items *which are in a specified region of the visualization* space are selected
4. The data items *within a graphically specified percentile of the view window (or data buffer)* are selected

Video of brushing modes: `http://vimeo.com/geirsmestad/iva-brushing`

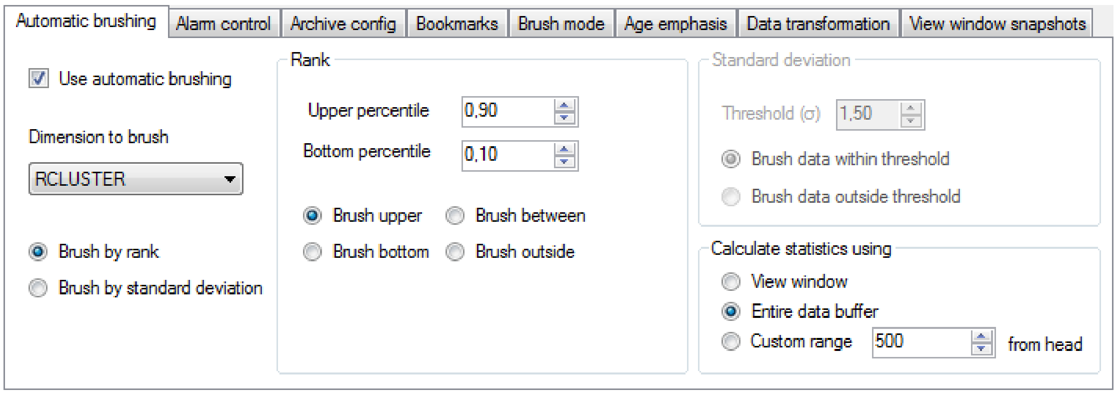
In the case of IVA of static data, all of these scenarios are identical. But given that new data items arrive continuously, and also that the range of the views might automatically change in response to new data items that are out-of-bounds, the distinction needs to be considered. We emphasize that brushing in a streaming data scenario is quite different from in a scenario of static data, as illustrated in the video above.

The first of these options is the most general and the default: The selection is updated every time a new data item leaves or arrives in the view window, and all views are updated according to whether or not each data item is within the data range initially brushed. This means, for example, that newly arriving data items can join the selection, if lying within the brushed data range. If the zoom levels of the views change, the data space range of the selection remains unchanged.

The second option selects each individual data item inside the brush at the time of brushing. As data items are purged from the data buffer or the view window position changes, the highlighted selection will change to reflect whether or not each of the selected items is currently inside the view window. Such a brush could be useful in order to e.g. mark a specific event in the data buffer or explicitly mark items to investigate their temporal position.

Option #3 is relevant whenever the zoom level of a view changes. Brushes are automatically updated to encompass all data items drawn in a specified portion of the

[4]We note that this list is not necessarily comprehensive; an arbitrary number of additional brushing scenarios can probably be defined.



[✓] Use automatic brushing

Dimension to brush: RCLUSTER

[● Brush by rank] ○ Brush by standard deviation

Rank

Upper percentile 0,90

Bottom percentile 0,10

[● Brush upper] ○ Brush between ○ Brush bottom ○ Brush outside

Standard deviation

Threshold (σ) 1,50

[● Brush data within threshold] ○ Brush data outside threshold

Calculate statistics using

○ View window [● Entire data buffer] ○ Custom range 500 from head

FIGURE 3.10: Control panel for automatic brushing, allowing the automated selection of data points based on the standard deviation or the rank.

screen. This also provides an implicit statistical selection if the zoom levels of the views *change automatically* in response to a change in the range of the data items. However, this option might otherwise be confusing and is mentioned mostly for its conceptual distinction.

Statistical brushing

Option #4 deserves some special attention. Depending on the plot type, the brushing operation implicitly marks a *percentile range* of the data items in the view. By looking at the values of the items encompassed by the brush and performing a statistical operation, these percentile ranges can be determined for each axis of the view. Whenever the contents of the view window are updated, this statistical operation can be performed to determine which of the new data items are within the initially selected range. We accomplish a brushing operation which is not possible on static data. This capability can be used to e.g. detect outliers. Statistical brushing is demonstrated in the video of brushing modes, at time 2:50 (linked from page 42).

Optionally, statistical brushing can be decoupled from the views. By manually specifying either a percentile range or a multiple of the standard deviation, we can specify a statistical brush whose selection is updated in tandem with the view window contents (interface illustrated in figure 3.10). In addition to providing a flexible method for highlighting trends and outliers in the data, this technique frees up the operator's attention since part of the brushing procedure is automated. Analyst attention is a valuable resource that we must conserve whenever possible.

Calculating statistics

For streaming data, we have three options to consider whenever we calculate statistical measures for a dataset[5]:

1. The data items in the current view window
2. The data items in the entire data buffer
3. All data items which have been received since connecting to the data stream [6]

An optional fourth option is to calculate the statistical metrics of the items currently marked by a data brush. Once again, the real-time domain introduces complexity which is hidden when we are working with a static dataset.

3.3.1 Compound brushing

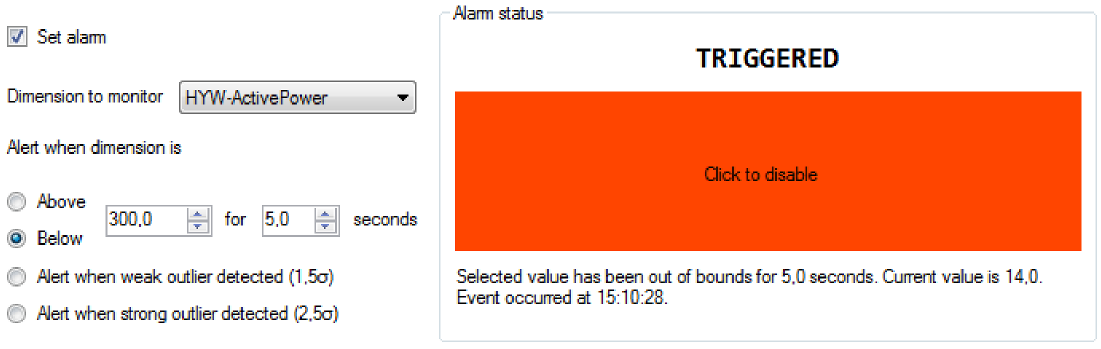
Second-level IVA, introduced on page 13, allows the possibility of using logical combinations of brushes (AND, OR, NOT) in order to refine a selection [64]. We have not implemented such a capability, but note that compound brushing is feasible within the constraints of time and attention that we have available during streaming data IVA. Provided a suitable user interface, compound brushing gives a simple but powerful enhancement to the basic Show & Brush loop, and could very well be implemented in a streaming data IVA interface.

3.4 Maintaining an overview of the present

We have described an IVA framework where the operator can choose to apply the Show & Brush loop on arbitrary portions of a live data stream. We have also discussed some ways in which the real-time aspect complicates the traditional IVA process. Absent from this discussion has been how to maintain an overview of what is going on in the present

[5]As for brushing, this list is not necessarily comprehensive, but these three options are meaningful in terms of the visualization system we have defined.

[6]Since there could be a limitation on the space available for the storage of historical data, these statistical metrics might have to be calculated using an incremental/in-place algorithm.


☑ Set alarm

Dimension to monitor: HYW-ActivePower

Alert when dimension is

Above [✓ Below] 300,0 for 5,0 seconds

Alert when weak outlier detected (1,5σ)

Alert when strong outlier detected (2,5σ)

Alarm status

TRIGGERED

Click to disable

Selected value has been out of bounds for 5,0 seconds. Current value is 14,0.
Event occurred at 15:10:28.

FIGURE 3.11: Alarm control panel, displaying a tripped alarm for low power output on a wind turbine.

when looking at data from the recent past. Human attention is largely undividable[7], and in most cases we must assume that the user does not know what will happen in the future. If there is only a single operator who is busy investigating events that happened in the recent past, the present is unattended. Events that take place during this process could require immediate intervention, e.g. if a temperature sensor goes out of bounds.

3.4.1 Alarms

The problem of sensor values that go out of bounds has already been solved in a variety of domains. A common approach is to tie an alarm to a sensor value, and interrupt the user's attention if erroneous or dangerous conditions occur (e.g. a temperature alarm on a common household refrigerator). A streaming data visual analysis framework should obviously include such a capability. In order to avoid alarms triggered by spurious outliers, we also include the ability to alert the user only if the selected dimension goes out of bounds for a selected amount of time (interface illustrated in figure 3.11).

In addition to providing alarms based on the violation of a specific threshold value, we allow the user to set an alert if the newly arriving data items vary dramatically from the data that have been observed so far. This provides the ability to set alerts for either absolute or relative conditions, which both have semantic significance[8]. Kehrer et al. [38]

[7]While humans usually have a certain ability to multi-task, there are often problems associated with performing even two simple tasks at the same time [65]. To which degree multi-tasking affects perception during real-time data analysis is not clear to us at this time, but the impact is certainly negative. We cannot in the general case assume that a human user will be able to monitor multiple visualizations with a degree of attention that matches a scenario where only a single visualization is monitored.

[8]For example, fluids undergo phase transitions at a specific temperature, which could require intervention. But a large increase in temperature can also indicate anomalous conditions which warrant immediate investigation.

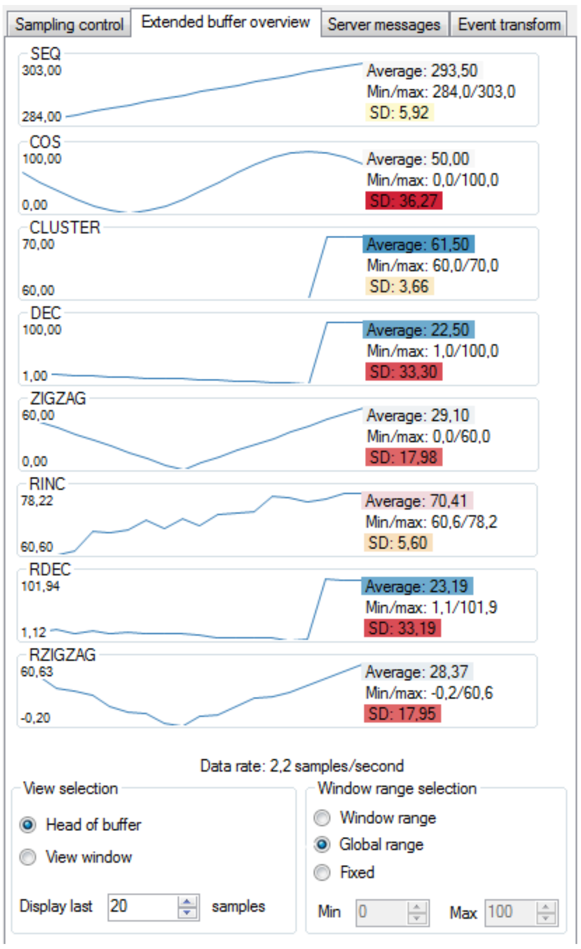
have previously suggested 1.5 standard deviations away from the mean as a distinction of a "weak outlier" and 2.5 standard deviations as characterizing a "strong outlier", so we use these numbers for our alarms. The statistical metrics must be calculated from a sufficiently large sample, in this case all the data items which have been observed. Due to the diversity of the data in different fields, alarm conditions are a likely candidate to be tailored to each individual application.

A side note: Expecting the unexpected

From a data management perspective, detecting sensor values that go outside of established boundaries is a simple task. Such events are in a sense expected, even though they are outside of the normal – detecting and monitoring abnormal behavior is a big part of the reason why data is monitored in the first place. Outliers or erroneous conditions that must be detected through mathematical operations such as statistical analysis are more difficult to handle, but are nevertheless within the category of "known unknowns". A much more insidious indication of error are *patterns* in the data that indicate something wrong, or *higher-dimensional relationships* between parameters, that are the result of erroneous operation. Such patterns are the main reason we use human analysts rather than automation, and are also what we intend to make easier to investigate by introducing better methods for streaming data visualization.

3.4.2 Overview visualization

We now return to the topic of how to let the operator maintain an overview of what is going on in the present while looking at older data. Traditional IVA allows for the flexible configuration of the views of the data. However, there is by default no mechanism for viewing all parameters at once. Such views could be created if necessary, but are excluded from the default analysis scenario for good reasons: Viewing a dense visualization of multiple heterogeneous parameters requires a high cognitive load (in addition to valuable monitor space). Limiting the focus of attention to the portions of the data that are currently expected to be relevant, provides a much better environment for reasoning, iteratively querying and investigating complex relationships in the data.



SEQ
303,00
284,00
Average: 293,50
Min/max: 284,0/303,0
SD: 5,92

COS
100,00
0,00
Average: 50,00
Min/max: 0,0/100,0
SD: 36,27

CLUSTER
70,00
60,00
Average: 61,50
Min/max: 60,0/70,0
SD: 3,66

DEC
100,00
1,00
Average: 22,50
Min/max: 1,0/100,0
SD: 33,30

ZIGZAG
60,00
0,00
Average: 29,10
Min/max: 0,0/60,0
SD: 17,98

RINC
78,22
60,60
Average: 70,41
Min/max: 60,6/78,2
SD: 5,60

RDEC
101,94
1,12
Average: 23,19
Min/max: 1,1/101,9
SD: 33,19

RZIGZAG
60,63
-0,20
Average: 28,37
Min/max: -0,2/60,6
SD: 17,95

Data rate: 2,2 samples/second

View selection
[✓ Head of buffer]
View window
Display last 20 samples

Window range selection
Window range
[✓ Global range]
Fixed
Min 0 Max 100

FIGURE 3.12: Enhanced Buffer Overview. For each dimension, we have the minimum and maximum values ever encountered on the left. The line plot is of the specified data, by default the head of the data buffer.

On the right, we display the average, minimum/maximum and standard deviation of the plotted data. To provide another visual cue, the average is color-coded to the position between maximum and minimum values ever encountered. Standard deviation is color-coded to the position between zero and the highest standard deviation ever encountered.

The average uses a diverging color scheme, the standard deviation uses a sequential scheme. We could choose different strategies for these cues depending on application and data characteristics.

In a real-time environment, we don't have the luxury of enough time to impose this limitation on the operator. Seeing at a glance what is going on in the present could be a requirement if something unexpected happens. We therefore introduce an overview visualization with line charts for all the dimensions of the dataset, where only the most recent data items are displayed. We dub this visualization "Enhanced Buffer Overview" (EBO). Since this view must be instantly available and not require excessive attention to configure, the Y axis limitations of the line charts can be set according to the most recent data range, the range of the entire data buffer or selected manually. In the two former cases, the range of each dimension is set individually. The interface is illustrated in figure 3.12.

Optionally, to increase the flexibility of this view, all dimensions of all the data items within the *current view window* can be displayed. This provides a quick overview of the current temporal focus, and could be of interest during general monitoring.

Many other schemes could be devised for providing an overview of the most current data.

For example, a scatterplot matrix would provide an overview of all the 2D relations in the streaming data.

3.4.3 Dynamic statistics/aggregation

Another option for displaying the current status at a glance, is to calculate and display some statistical metrics for each dimension. We choose to calculate the minimum and maximum, average and standard deviation of each dimension and display these values alongside each line graph in the EBO. (The metrics are calculated using only the values visible in the EBO).

The overview visualization we have realized so far, could be further enhanced to improve its perceptual aspects. A line graph is a dense but naïve technique to represent information, and has perceptual drawbacks when displayed in a dense visualization. A pseudo coloring technique, such as two-tone pseudo coloring [69], would likely be an improvement in this regard. Unfortunately, time constraints prevented us from fully exploring this avenue. Visualizing how the statistics develop over time, is also an option which could later be explored in detail.

3.5 Handling high data volumes

Human perception is limited when it comes to real-time observation and interaction. Card et al. [10][9] mention three levels of interaction, at roughly 0.1 seconds, 1 second and 10 seconds, called the *human time constraints*[10]. These intervals define the limit to smooth perception, reaction/dialogue and a single *interaction cycle*, respectively. In other words, there are limits to how fast different types of human interaction can happen, provided that the person is expected to have a complete overview of what is going on. According to this theory, percepts that occur at smaller intervals than approximately 0.1 seconds are perceived as simultaneous. This places a limit to the data rate a human analyst can fully monitor. In addition, an analyst will be unable to *react* to distinct events that take place less than a few seconds apart, and will be unable to perform

[9]See Chapter 3, Interaction.

[10]The authors did not claim that these are absolute limits, but they are within the correct order of magnitude for each level of interaction.

detailed analysis tasks for events that occur with smaller intervals than a few tens of seconds.

The human time constants are easily challenged in a real-time data analysis scenario, and we need to introduce techniques to help alleviate this limitation. There are two conceptual ways that the human time constants can be challenged in a real-time application: Either the *data rate* is higher than the 1-10Hz defined by the perceptual interaction limit, or the *variability* in the data is faster than the 10-30 second limit defined by the interaction cycle. These scenarios are not mutually exclusive.

3.5.1 Data rate reduction

The simplest way to alleviate the problem of a high data rate, is to reduce the data rate. This necessarily causes a loss of information, which could be large. Operator judgement is important. There are many ways to implement such a functionality:

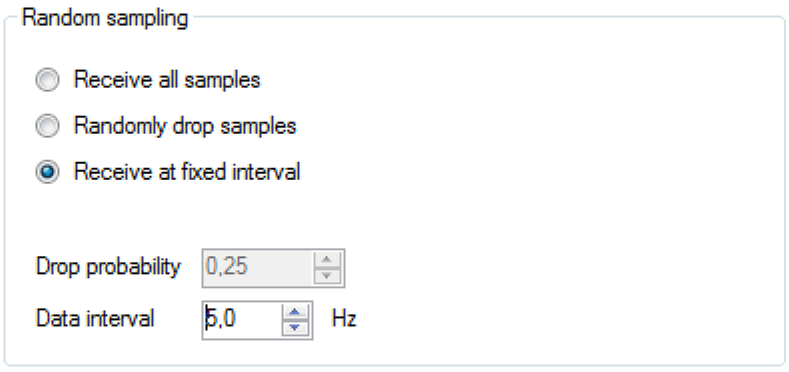
- Samples could be randomly dropped
- Data items could be sampled from the data stream only at a fixed interval
- Samples could be reconstructed, and the interpolated value added to the data buffer only at a fixed interval

We have implemented the first two of these methods (figure 3.13), which are examples of statistical sampling [14]. We discuss interpolation in greater detail below. We consider the second option better than the first, since it avoids the extra cognitive load of a variable data rate.

A drawback with either of these methods is that minima and maxima in the data stream are not preserved. These could have important semantic value. Minima- and maxima-preserving sampling schemes could be devised if necessary.

3.5.2 Frequency-based visualizations

Another way to sidestep the issue of a high data rate, is to use visualizations that are frequency-based rather than item-based. We implement histograms to demonstrate this


Random sampling

- Receive all samples
- Randomly drop samples
- [✓ Receive at fixed interval]

Drop probability 0,25

Data interval 5,0 Hz

FIGURE 3.13: Control panel for data rate reduction by random sampling

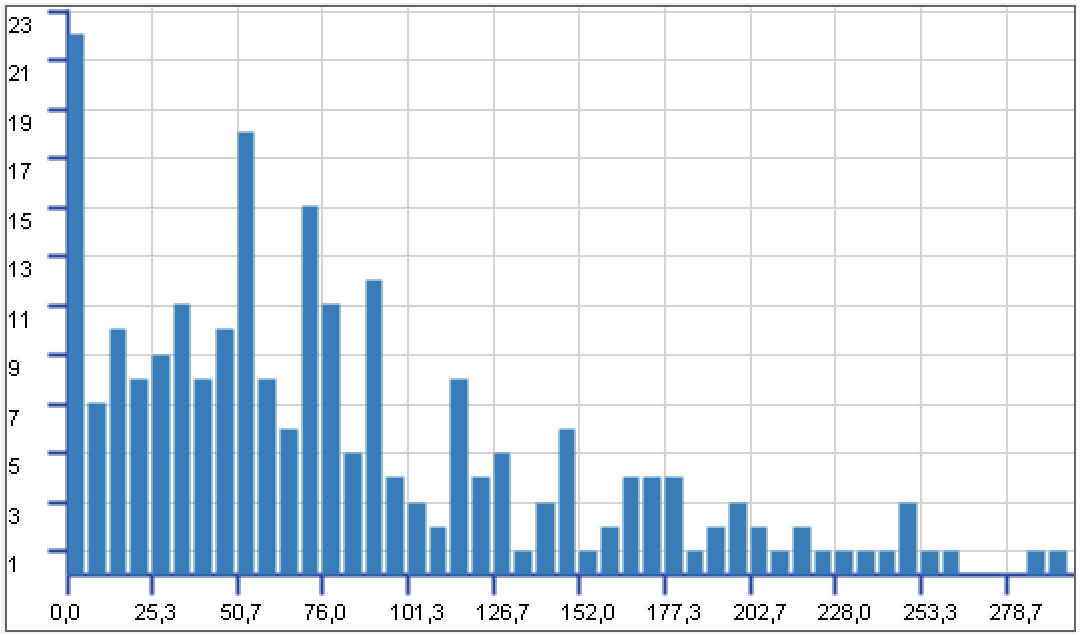


FIGURE 3.14: Sample histogram. The histogram is a well-known frequency-based visualization technique. This particular histogram displays power produced (kW) by the Hywind wind turbine during a certain time span.

capability (see figure 3.14). Another possibility would be to use Kernel Density Estimation [48], which estimates the probability density function implied by the distribution of the data.

Frequency-based visualizations provide the benefit of not visualizing each data item individually. In addition to alleviating problems with temporal perception, such visualizations have the added bonus of avoiding overplotting, which plagues many item-based visualizations when data volumes are high [44]. Data with high variability (e.g. alternating between low and high values in a short interval) could still pose a challenge wrt. the human time constants.

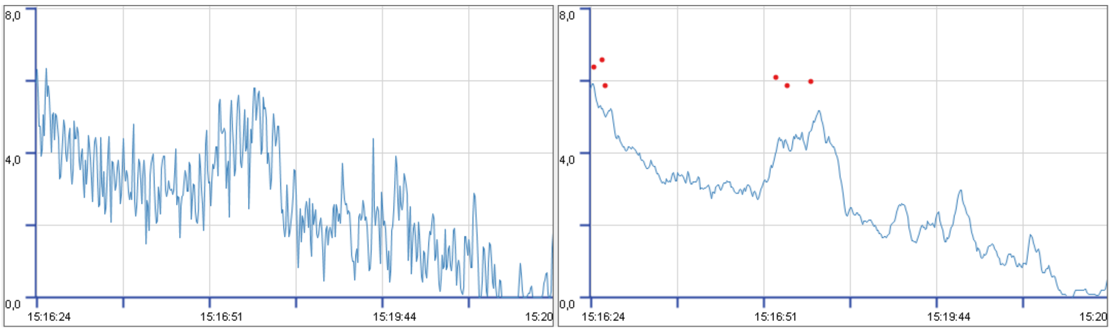


FIGURE 3.15: Comparison of unfiltered wind speed readings (left) and trend+outlier view (right). The trend+outlier view is a simple moving average with window size 9. Data items more than 2.1σ outside of the mean are highlighted in red[11]. The charts contain 350 sensor readings.

3.5.3 Trend + outlier views

A second approach is to reduce the *variability* in our data through aggregation. One obvious choice for such a transformation is the *moving average*, which can in the terminology of signal processing be considered a convolution. Averaging can help to smooth out noisy data in order to reveal trends that are not obvious when looking at the raw data. The cost of this operation is obviously that we lose the minima and maxima, in addition to hiding the details of the individual data values.

A diplomatic middle ground is to create a "trend + outlier view". In addition to displaying the moving average in a line chart, we continuously calculate the mean and standard deviation of the data buffer. Individual data values that deviate from the mean more than a specified multiple of the standard deviation, are highlighted next to the moving average. This view is illustrated in figure 3.15.

We have chosen the *simple moving average* for this calculation. For elements x of the view window, the value of the moving average of degree k at position n is given by

$$A_n = \frac{x_n + x_{n\text{-}1} + x_{n\text{-}2} + \cdots + x_{n\text{-}k\text{-}1}}{k}$$

The simple moving average has the disadvantage of shifting the averaged values with regards to the original data. But unlike the central moving average, it can be computed for the whole data buffer without making assumptions about data values that have not

[11]In the trend+outlier view, the definition of an outlier can be specified in the user interface. For this particular example, a value of 2.1σ away from the mean fit well with the data. The definition of an outlier usually depends on the dataset.

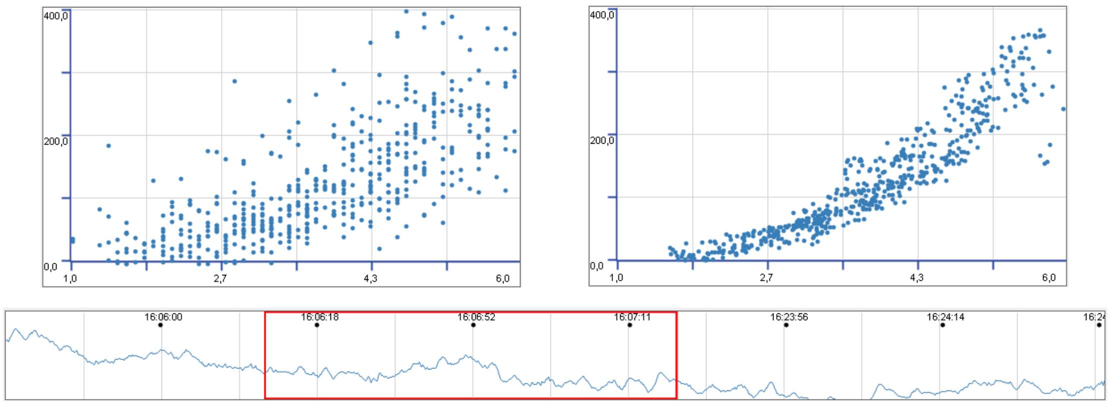


FIGURE 3.16: Scatterplots of wind speed vs. produced power in kW. Data has been smoothed with the moving average across the whole data buffer, as indicated by the smoothed wind speed plot in the buffer view. The left-hand scatterplot displays the raw, unfiltered data.

yet been received. To make up for this compromise, we shortly consider the application of convolution, which allows the central moving average to be easily computed.

3.5.4 Generalized signal processing

The trend + outlier view is a local aggregation technique, transforming only the current view of our data. The view type is limited to the line chart. Extending the notion of aggregation, we can also allow the transformation of the entire data buffer (see figure 3.16). The user can selectively apply transformations to any of the four views in the visualization interface. This allows for a generalized comparison of aggregated and raw data in any of the three main view types that we have implemented.

We implement the simple moving average as defined above, in addition to the *simple moving variance* (SMV), illustrated in figure 3.17. The moving variance measures how much a sequence of values varies over each interval. For each position n of the data buffer, it is defined as follows, using the same constraints as for the moving average:

$$V_n = \frac{\sum_{i=n-k-1}^{n}(x_i - \mu)^2}{k}$$

where μ denotes the average of the current selection, i.e. $\mu = \frac{\sum_{i=n-k-1}^{n}(x_i)}{k}$ for each position n of the data buffer.

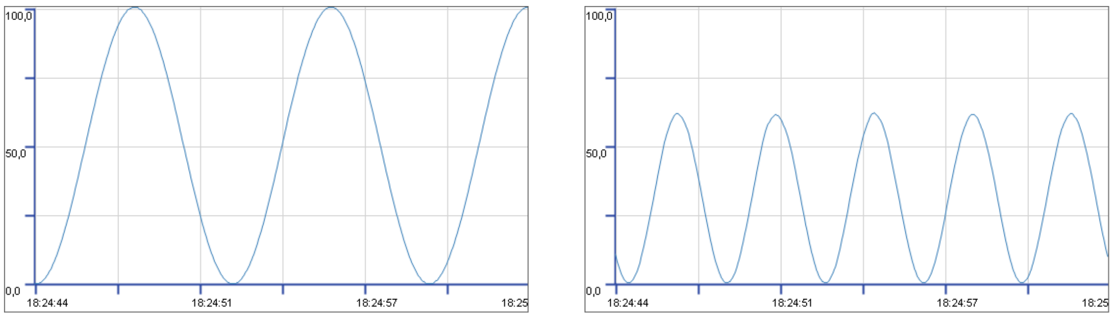


FIGURE 3.17: The simple moving variance (SMV) of a sine curve. Original data on the left, variance on the right. This curve contains 180 data points, and the SMV has degree 9. Note that the SMV has been calculated for the entire dataset, while these plots only show the current view window. Since the SMV will be slightly shifted in relation to the original data, the leftmost part of the SMV plot incorporates some data points which are not visible in the plot of the original data.

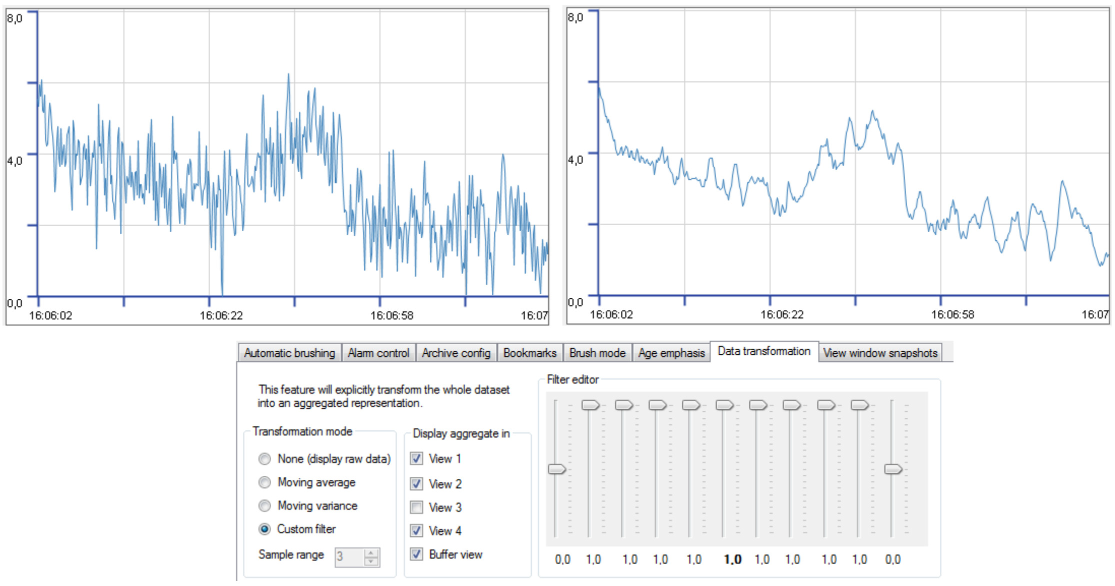
This leads us to the notion of *generalized signal processing* on a real-time data source. Using convolution, we can create an interface for user-directed linear filtering on any dimension of the dataset (illustrated in figure 3.18).

If we consider each dimension of our data buffer to be a real-valued function f and our convolution filter a function g, both defined on the set of integers $\mathbb{Z}$, the discrete convolution of a data buffer of length k is given by

$$(f * g)[n] = \sum_{m=0}^{k} f[m]g[n-m]$$

calculated for each position n along the data buffer. Values outside of the data buffer are defined to be equal to the outermost value[12]. For simplicity, we let $m = 11$ and g $\in$ [-1, 1]. We define an interface where the user can select the 11 values of g, and let g equal zero outside of [-5, 5]. This visual convolution selector is interpreted from right-to-left by the software, in order to match the intuitive picture of the convolution function as a filter which sweeps across the data buffer during the transformation. To ensure a uniform scale on the final signal, we also divide the result during each step by the absolute value of the definite integral of g. (In the case of a zero-valued definite integral, we divide by 1 instead, which corresponds to not scaling the result).

[12]In principle, there are many different strategies for handling the edge values when calculating a discrete convolution. We selected the outermost value for the technical simplicity of this approach, but note that more sophisticated schemes are possible.



This feature will explicitly transform the whole dataset into an aggregated representation.

Transformation mode
- None (display raw data)
- Moving average
- Moving variance
- [✓ Custom filter]

Sample range 3

Display aggregate in
- [✓] View 1
- [✓] View 2
- [ ] View 3
- [✓] View 4
- [✓] Buffer view

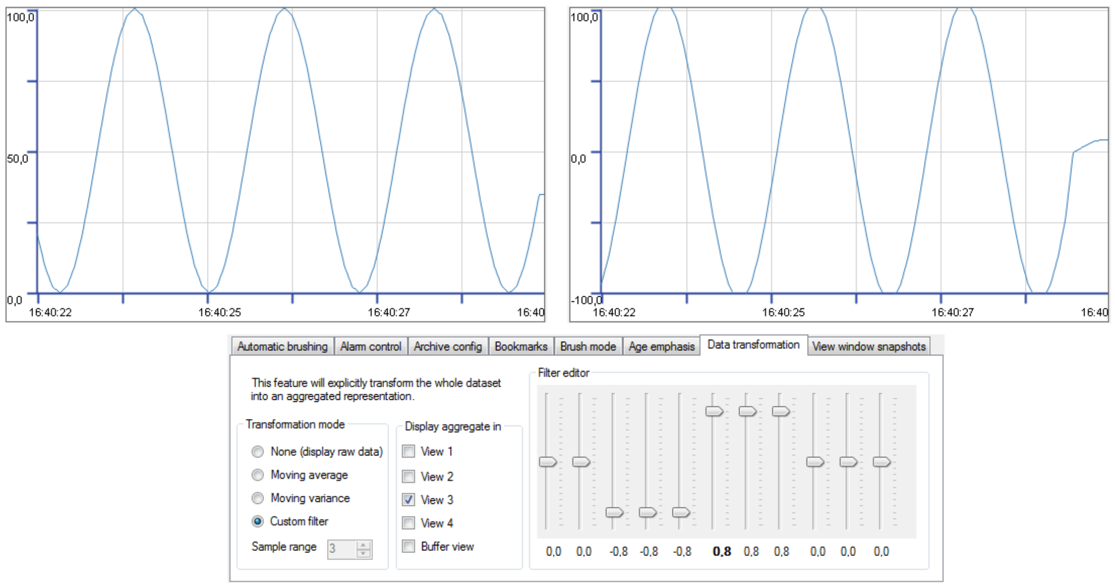
0,0 1,0 1,0 1,0 1,0 **1,0** 1,0 1,0 1,0 1,0 0,0

FIGURE 3.18: Moving average by generalized linear filtering. Similar data to the example used earlier, but this time processed generally through a *box filter* rather than an algorithm that specifically calculates the moving average.

Using generalized linear convolution, a skilled operator can interactively and in real time apply a wide range of linear filters to her data stream. This provides a flexible tool for transforming noisy or fluctuating data streams to a form which better matches the human operator's requirements and perceptive capabilities, in addition to providing a completely new tool for real-time data transformation. For instance, estimating the numeric derivative of a dimension is accomplished by creating a filter that approximates the central difference operator. This operation is illustrated in figure 3.19. Allowing different filters for different dimensions would further improve the capability of this approach.

Using data transformations to allow more complex real-time IVA

Konyha et al. define four levels of complexity in Interactive Visual Analysis [42]. This thesis is mostly concerned with the first level, under the assumption that there will not be enough time to allow an operator the cognitive load and physical actions necessary to e.g. perform attribute derivation or application-specific feature extraction on the live data. However, as we have seen in section 3.3.1 and in the previous paragraph, both the second and third levels of IVA can be adapted to real-time applications: Linear filtering



This feature will explicitly transform the whole dataset into an aggregated representation.

Transformation mode

( ) None (display raw data)
( ) Moving average
( ) Moving variance
(●) Custom filter

Sample range 3

Display aggregate in

☐ View 1
☐ View 2
☑ View 3
☐ View 4
☐ Buffer view

Filter editor

0,0 0,0 -0,8 -0,8 -0,8 **0,8** 0,8 0,8 0,0 0,0 0,0

Figure 3.19: Numeric differentiation by realizing a central difference operation. Differentiation is implemented through the use of a gradient filter. The dataset is a sine curve (scaled to an amplitude of 100), displayed filtered on the right and unfiltered on the left. The filtered curve corresponds to the cosine, as expected[13].

is a transformation of the original data, which is the requirement for an IVA interaction to be considered third-level.

When performing transformations on our data, there are two separate issues that slow down the process: The cognitive load required to plan the transformation, and the user interface actions needed to apply it to the data. There is little to be done about the cognitive aspect, but our observations of existing IVA methodology indicate that there are great opportunities for simplifying the user interface tasks. Often, data transformations are performed with external tools directly on a dataset in CSV or XLS format. Integrating the process with the visualization interface through e.g. an interactive formula editor should greatly expedite the process.

It follows from these observations that moderately complex attribute derivations should also be feasible during real-time IVA, i.e. calculating new dimensions of the dataset based on the other dimensions. To do this, we would have to create a formula editor which is sufficiently interactive and does not interfere with the real-time monitoring.

[13]Note that the amplitude of the result is affected by the scaling factor. As described above, the scaling factor is now 1, since the definite integral of g is zero when g corresponds to the central difference operation. We have therefore scaled the result manually, by setting the weights of g to 0.8 instead of 1.0.

3.5.5 Intermittent view updates

Video of intermittent updates: `http://vimeo.com/geirsmestad/iva-intermittent`

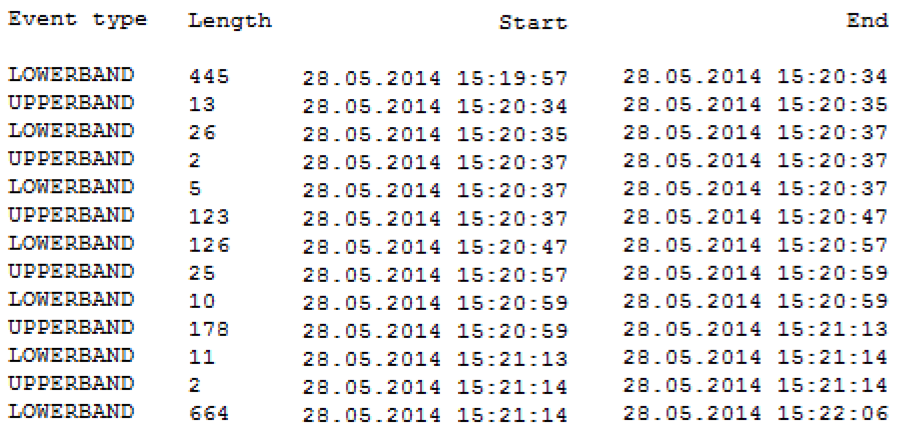
A third avenue for handling high data rate and high data variability is to limit the *view update rate* to correspond to the human analyst's perceptual limitations, as described on page 48. In other words, we would like to keep our views static except for occasional animated updates to make the views match the current data (illustrated in the video above). If we do this, we are in effect transforming our real-time data analysis application into an intermittently static data analysis application. This has the expected consequence of introducing a small delay between the observed and the current data values, but this could be an acceptable tradeoff.

We allow the user to define a view update interval and an update speed. In intermittent mode, views are not updated continuously as new data items are arrive. Instead, the views are frozen during the update interval, which corresponds to the ”interaction cycle” length in human perception, for example 10-30 seconds. New data items received during this interval are stored in a temporary buffer. At the end of the interval, all views are updated with the data items that are presently inside the view window. In order to avoid change blindness, this update is animated in the order that the new data items were received. The length of this animation should correspond to the ”reaction/dialogue” length in human perception (on the order of one second).

This view update scheme largely eliminates the *perceptual* difficulties of high data rates and variability, but clearly introduces some complications. The abovementioned delay before the present data is displayed is one aspect. A bigger issue is that decoupling the views from the real-time data stream could cause the operator to lose some of the cognitive reference to the temporal order of new events. This problem also occurs when the user is looking at older data rather than the present, and we investigate a technique to alleviate this problem at the end of this chapter.

3.5.6 Event transformation

Aggregation in the form of averaging or linear filtering is a straightforward way of processing heterogeneous data for easier human comprehension. A different approach



```
Event type   Length              Start                      End

LOWERBAND    445      28.05.2014 15:19:57     28.05.2014 15:20:34
UPPERBAND    13       28.05.2014 15:20:34     28.05.2014 15:20:35
LOWERBAND    26       28.05.2014 15:20:35     28.05.2014 15:20:37
UPPERBAND    2        28.05.2014 15:20:37     28.05.2014 15:20:37
LOWERBAND    5        28.05.2014 15:20:37     28.05.2014 15:20:37
UPPERBAND    123      28.05.2014 15:20:37     28.05.2014 15:20:47
LOWERBAND    126      28.05.2014 15:20:47     28.05.2014 15:20:57
UPPERBAND    25       28.05.2014 15:20:57     28.05.2014 15:20:59
LOWERBAND    10       28.05.2014 15:20:59     28.05.2014 15:20:59
UPPERBAND    178      28.05.2014 15:20:59     28.05.2014 15:21:13
LOWERBAND    11       28.05.2014 15:21:13     28.05.2014 15:21:14
UPPERBAND    2        28.05.2014 15:21:14     28.05.2014 15:21:14
LOWERBAND    664      28.05.2014 15:21:14     28.05.2014 15:22:06
```

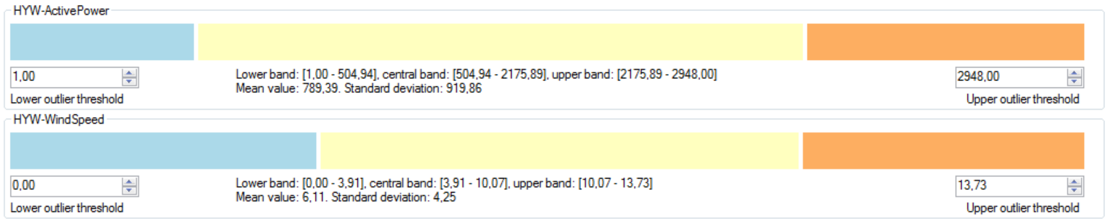
FIGURE 3.20: Sample of the event log for a single dimension of the data stream.

is to transform the data stream into a sequence of *events*, where each event attempts to capture a semantic aspect of the data. The space of possible event transformations is unbounded and could be heavily application-dependent. Aigner et al. have discussed this subject in detail [2]. We describe one event specification which has some general applicability.

For each dimension of our data source, we define five data value limits: *Lower outlier, lower band, central band, upper band* and *upper outlier*. These limits assign every data value to one of five bins. We consider a continuous sequence of data items within the same bin to be one *event*. E.g. for a pressure reading, we consider 30 readings in the central band, followed by five readings in the upper band, followed by one reading in the upper outlier band to be three separate events of length 30, 5 and 1, respectively. Events are parsed from the data stream as they occur and stored in a sequential list, one for each dimension of the data stream (see figure 3.20). This scheme is a run-length encoding on the values of the data.

We define a user interface which allows the operator to automatically set initial band limits based on the current contents of the data buffer. The initial bands correspond to the limits defined in section 3.4.1: data within $\boldsymbol{\mu \pm 1.5\sigma}$ are in the central band, data outside of the central band but within $\boldsymbol{\mu \pm 2.5\sigma}$ are in the upper or lower band, and data outside of $\boldsymbol{\mu \pm 2.5\sigma}$ are upper or lower outliers.

These initial bands may not correspond to the best categorization, and we therefore allow the operator to manually adjust these limits. The interface for event band selection is illustrated in figure 3.21. Selecting the correct bands is important in order for the


HYW-ActivePower

1,00
Lower outlier threshold

Lower band: [1,00 - 504,94], central band: [504,94 - 2175,89], upper band: [2175,89 - 2948,00]
Mean value: 789,39. Standard deviation: 919,86

2948,00
Upper outlier threshold

HYW-WindSpeed

0,00
Lower outlier threshold

Lower band: [0,00 - 3,91], central band: [3,91 - 10,07], upper band: [10,07 - 13,73]
Mean value: 6,11. Standard deviation: 4,25

13,73
Upper outlier threshold

FIGURE 3.21: User interface for event band selector, displaying initially calculated bands that have been tweaked by the operator.

event transformation to be a proper simplification of the data stream. If the event count per time unit is high, viewing an event display will be equally demanding but less informative than looking at a more direct representation. For every actual data source, domain knowledge will provide specific semantic meaning for the different ranges of the data. This knowledge should be used to select more specific event bands. Sensor ranges that indicate abnormal operation or failure modes are of particular interest.

3.5.7 Event overview

We create three different views of the event list. The first provides a dense overview of all dimensions, where color corresponds to event type and width corresponds to event length (see figure 3.22)[14]. There is a 1-to-1 relationship between time and width. Preattentive perception ensures that abnormal data values, the time at which they occurred and any direct relationship with abnormalities in other dimensions, are visible at a glance. The Relaxed Event Timeline of Fischer et al. is very similar to this approach, though it also incorporates space compression for older events [21].

3.5.8 Item-based event visualization

The 1-to-1 relationship between X axis dimension and time is problematic if we have a long monitoring session, are interested in the temporal difference between long events or mostly want to monitor the present while keeping recent history as context. We therefore devise a pure item-based representation where each event has a fixed width. Event height is scaled according to the length of the event, and the vertical *position*

[14]This visualization is still continuous with regards to the discretized event data, and hence does not fully incorporate the discretized nature of events. It is still a useful visualization for viewing the relationship between the continuous data and the discrete event representation. The equidistant time mapping on the X axis also makes it easy to see at a glance when each event begins and ends, and to compare events from different parameters.

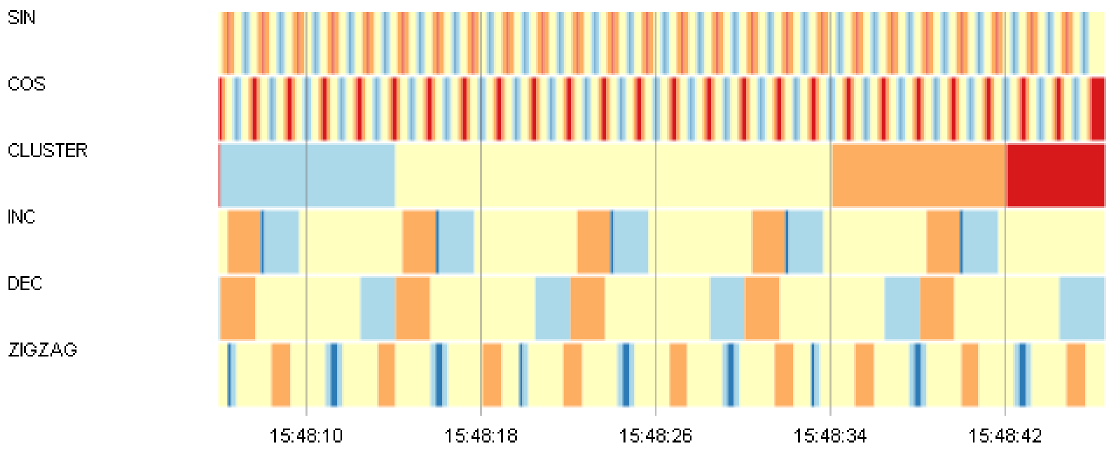


FIGURE 3.22: Event overview for six dimensions of a synthetic test dataset. Each pixel corresponds to the length of a single data point, each color corresponds to one of the five bands we have defined.

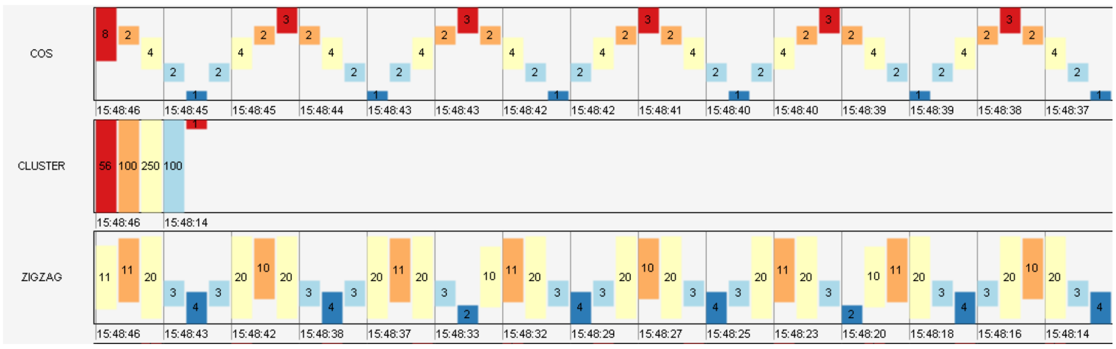


FIGURE 3.23: Item-based event view. The current event is always on the left-hand side[15]. Old events are gradually shifted to the right as new events occur. Event length is printed on each event bar. This is also a synthetic dataset, used for illustrative purposes.

of each event is selected according to event type. This means that we are using both color and position as cues for the event type, which eases the cognitive load during monitoring. New events are inserted from the left by shifting older events to the right (see figure 3.23).

This scheme decouples each dimension along the X axis, but the *present events* are always aligned in the leftmost part of the view. Timestamps and the specific length of each individual event can be displayed as required.

[15]Inserting new events on the left instead of on the right, is inconsistent with the convention used in the rest of this thesis. We initially designed the view with new events inserted on the right, with older events shifted to the left as the view is filled. However, a technical issue with regards to our user interface library prevented this solution from giving a good user experience. The image presented here has new events inserted on the left, but we note that this is not the ideal choice.

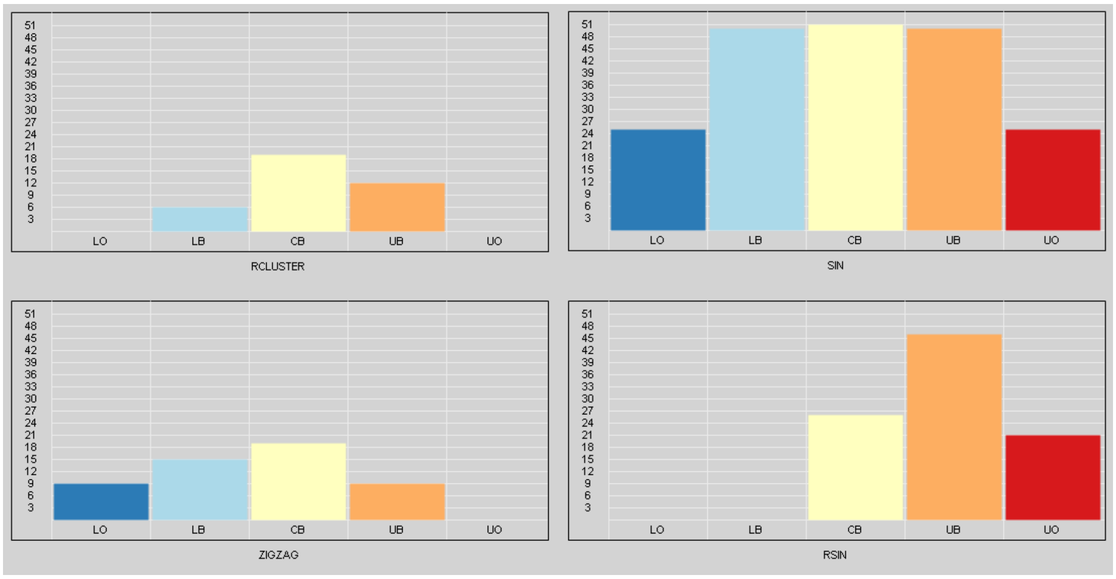


FIGURE 3.24: Simple event histogram of synthetic test dataset.

Since space along the Y axis is limited but event length is unbounded, event height must be limited. We calculate the height of each event by exponentially approaching the total height available to each dimension:

$$h = \text{totalHeight}\left(1 - \text{base}^{\text{eventLength}}\right)$$

where *base* $\in \langle 0,\ 1\rangle$ and *eventLength* $\in \mathbb{N}$. Events in the central band are always vertically centered and outliers always anchored to the top or bottom of the view. Upper and lower band events are positioned so the distance to the bottom is three times higher than to the top (and vice versa). *Base* must be adjusted according to the expected event lengths. In our examples where most event lengths are smaller than 50, a *base* value of 0.95 yields a good scaling for the bar heights. Higher event count requires a *base* value closer to 1.

3.5.9 Frequency-based event visualization

Finally, we define a qualitative (rather than quantitative) event view which provides a high-level overview of the event distribution and characteristics, while only having a loose relationship to each individual event.

We begin by drawing histograms of the events within each dimension (figure 3.24). An abnormal distribution will be immediately apparent, which means that this view could serve as part of a ”status overview” if we display e.g. all events that have occurred in the last 24 hours. We note that since there are always five bars in each histogram, the horizontal space is poorly utilized. We can use this space to provide an indication of the distribution of event lengths.

One intuitive solution is to separate the events for each dimension into lists according to event type, sort each of these lists in descending order and ”stack” the events on top of each other as we draw each histogram bar. With event length scaled to bar width similarly to the item-based view, each histogram bar will be a ”pyramid” with the width of each level corresponding to event lengths.

However, we wish to decouple the visualization from the individual events. We therefore choose a scheme based on *resampling* the event lengths. We sort the events of each histogram bar by event length as described above. Using these five lists, we define a function $\boldsymbol{L(p)}$ of event length as a function of position in the sorted event list. $\mathbf{L}$ is defined on the set of real numbers $\mathbb{R}$, with linear interpolation between event lengths for non-integer values of p.

Finally, when drawing the histogram bars, we choose a resampling interval i corresponding to a vertical number of pixels. If i is set to 1 and is smaller than the height corresponding to a single event, the scheme will be identical to the simpler technique described above. Each histogram bar is modified in width at each multiple of the resampling interval, with the width calculated as

$$w = \text{totalWidth}\left(1 - \text{base}^{\boldsymbol{L(y)}}\right)$$

where *base* $\in \langle 0, 1\rangle$, as previously. y is selected in each step such that it corresponds to the current multiple of the resampling interval, scaled by the pixel height of each event. To make the visualization more visually pleasing, we draw each histogram segment as a trapezoid rather than a rectangle. This causes significant imprecision only at high resampling intervals. The final result is illustrated in figure 3.25.

As indicated above, this scheme provides a qualitative indication of the distribution of event lengths within each event category. It will be immediately obvious whether the

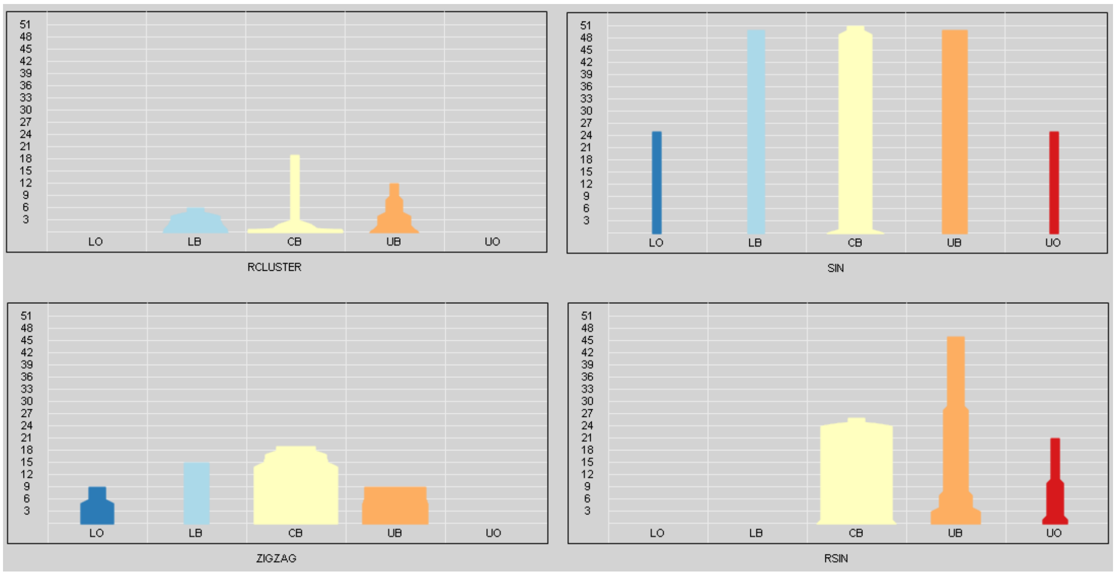


FIGURE 3.25: Enhanced event histogram. The data is identical to the plain histogram in figure 3.24. All dimensions except RCLUSTER are dominated by long events in the central band. SIN is distinguished by a large number of short events in the outlier bands.

cohort consists of short or long events, while keeping the visualization consistent whether each cohort represents only a few or thousands of events.

This concludes the discussion of how to improve operator control in the face of high data rates and heterogeneous data.

3.6 Mental map preservation

Most of the common visualization types are not designed for continuous updates. When adapting static visualizations to live updates, a number of complications occur. Many of these are related to *loss of user context* and *mental map invalidation*, as detailed by Krstajić and Keim [44].

Although we have chosen plot types that minimize the loss of context and mental map invalidation when updated with new data, the user has no control over what data may be received in the future. New data could either be outside of the range currently displayed (leaving the user oblivious to the new development) or old could leave the view window such that the data is drawn only in a small portion of the view. This suggests that we should be able to automatically change the boundaries of the views to adapt to the current range of the data.

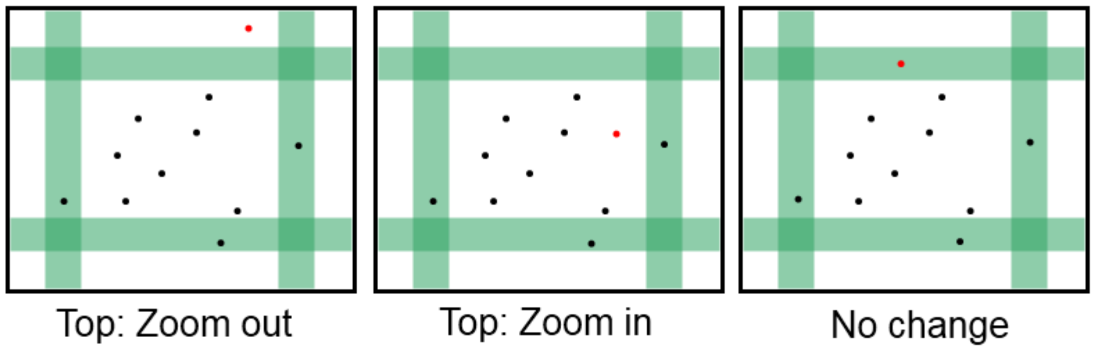


FIGURE 3.26: Illustration of the three cases for automated zooming, demonstrated on the top edge. Similar cases hold for the left, right and bottom edges. Peripheral regions in green, although only the top region is relevant to this example. The red data point is the newest addition to the plot.

3.6.1 Automated zooming

Implementing such an automated zooming feature requires that we avoid some pitfalls:

- Abrupt changes in view dimension exposes the user to *change blindness* and also breaks the user's mental mapping of view distances to data ranges
- Updating the view dimensions too often, e.g. in response to small changes in the data range, will require the user to constantly update his or her mental map. This is visually confusing.

These observations indicate that we should implement a *hysterisis-based* update scheme so that the view dimensions are updated only when necessary, and that these updates should be gradual in order to avoid change blindness.

The hysterisis scheme we have chosen defines a central region of the plot, along with a peripheral region along each view edge. View dimension updates are calculated every time a new data point is received. For each view edge, we zoom *out* if the newly received data point is *outside* of the peripheral region, and we zoom *in* if the peripheral region is empty and there are no data points outside of it. This ensures that all data points are displayed in the center of each view[16], while the zoom level is not changed if the new data points only deviate slightly from the old.

[16]This assumes that data items are somewhat evenly distributed, which will not necessarily be the case. However, this simplification is necessary as long as we use linear coordinate axes.

How do we set the new view limits once we determine that an update is required? Assuming that the peripheral region is defined by data space coordinates set to some fraction of the view dimensions, we create four equations that determine the edge coordinates of the view in the data space. The key restriction is that after resizing the view, the outermost data items for each dimension should be in the center of the corresponding peripheral region (see figure 3.26). The equation for the top of the scatterplot follows; the reasoning is similar for the three remaining edges and the other view types.

Video of automatic zoom: http://vimeo.com/geirsmestad/iva-autozoom

Let k be the data space Y coordinate of the topmost data item, *outLimit* and *inLimit* be the Y coordinates delimiting the peripheral region after resize and *maxY* and *minY* be the coordinates of the top and bottom edges of the view after resizing. We get

$$k = \mathrm{inLimit} + 0.5\,(\mathrm{outLimit} - \mathrm{inLimit})$$

Let *outMargin* and *inMargin* be the data space distance from the top of the view to the top and bottom of the peripheral region, respectively. Further, O and I are the multiples of the total view height that these numbers correspond to. We get

$$\begin{aligned} k &= 0.5\,(\mathrm{maxY} - \mathrm{inMargin} + \mathrm{maxY} - \mathrm{outMargin}) \\ &= 0.5\,(\mathrm{maxY} - (\mathrm{maxY} - \mathrm{minY}) \cdot \mathrm{I} + \mathrm{maxY} - (\mathrm{maxY} - \mathrm{minY}) \cdot \mathrm{O}) \\ k &= \mathrm{maxY} - 0.5 \cdot \mathrm{I} \cdot \mathrm{maxY} - 0.5 \cdot \mathrm{O} \cdot \mathrm{maxY} + 0.5 \cdot \mathrm{I} \cdot \mathrm{minY} + 0.5 \cdot \mathrm{O} \cdot \mathrm{minY} \end{aligned}$$

$$\Leftrightarrow \mathrm{maxY} = \frac{k - 0.5 \cdot (\mathrm{O} + \mathrm{I}) \cdot \mathrm{minY}}{1 - 0.5 \cdot (\mathrm{O} + \mathrm{I})} = \frac{2\mathrm{k} - (\mathrm{O} + \mathrm{I}) \cdot \mathrm{minY}}{2 - (\mathrm{O} + \mathrm{I})}$$

Hence, provided the value k of the outermost data item and multiples I and $O \in \langle 0,\, 1 \rangle$ of the view height denoting the dimensions of the peripheral region, we can calculate the

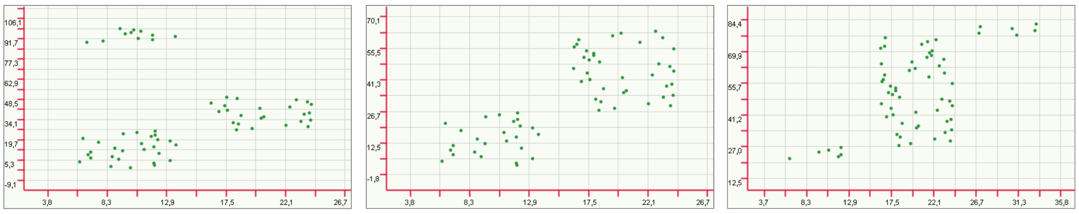


FIGURE 3.27: Three successive instances of automated zooming during a live data analysis session. Data arrives on the right, and leaves from the left and the top. Note that the hysterisis feature always preserves some room on each side of the data items, as well as the background reference lines that are always fixed to the data space.

required top edge of the view after resizing. Three similar equations govern the bottom, left and right edges of the view.

These four equations are actually pairwise of two unknowns - the top and bottom edges of the view, and the left and right edge of the view. We solve these numerically, by iteratively inserting the currently known values until the result converges. For practical purposes, this is sufficient: The image drawing operations dominate the CPU budget of our application, and numeric operations represent only a small part of the run-time calculations. We also only require the numerical accuracy to be within 0.5 pixels of the exact value. If greater speed or the exact value is required, the equations can of course be solved explicitly. The final result is illustrated in figure 3.27, and in the video above.

Preserving context

In order to avoid the risk of change blindness, the resize dictated by the conditions above must be gradual. We let the resizing happen at a smooth framerate over a time interval selected by the user, by default on the order of one second. In addition, the resizing also encompasses a number of *background reference lines* drawn behind the data items. These reference lines stick to the scale markings on the side of the view, providing a visual cue that imitates the effect of physically moving in relation to the observed objects. This adds an additional cue to the resizing operation, reducing the mental effort required to keep track of the change.

On the subject of context preservation, there is one other consideration we should mention. An observant reader will notice that the numerical scale markings on the vertical axis in the previous examples changed as the view shifted to the right. This is caused by a decision in our particular subroutine that draws the numeric scale only on every other tickmark. In this particular example, the ticks that get the numeric scale flip around during the zooming procedure. In addition, minor inconsistencies in the scale markings occur as we zoom, on the order of 1%. This is caused by our choice of image-order drawing on the scale markings - a seemingly insignificant choice which is perfectly reasonable when views are static, but becomes subject to sub-pixel inaccuracy when the views are animated. We elected to preserve these inconsistencies in order to highlight the issues.

Final notes on handling high data volumes

In the beginning of this chapter, we only briefly mentioned the subject of automatic view configuration. Setting the range parameters automatically is a requirement for allowing the user to direct her attention to the analysis tasks rather than managing the quirks of the software. The technique described here is one way to implement such automatic configuration, which is important regardless of whether the data is static or dynamic.

3.6.2 Using histograms with real-time data

Video of animated histograms: http://vimeo.com/geirsmestad/iva-histograms

The humble histogram provides some challenges when adapted to real-time data. A naïve histogram implementation lets the user select the number of bins and then divides the available data into the requested number of bins. This approach does not work when the data changes during analysis. When new data items are received outside of the current range, the data width and data position of each bin will change. Since the underlying visualization stays constant in width, there is a sharp break between the data and the visualization (and hence our underlying mental model). To alleviate this problem, we instead configure the histogram by *bin size*, the data range encompassed by each bin. Using this approach, width will always be fixed to data range. The number of histogram bars will grow or shrink as the range of the dataset changes over time.

Automated zooming in the histograms is implemented with a similar technique as for the scatterplots, although the equations are simpler since the bars are always anchored to zero at the bottom of the view. Bin sizes are not changed (in terms of the data space) during automatic zoom. Automatically changing bin size could be necessary if large zooming operations are common, but unfortunately time pressure prevented us from exploring this. We note that such an adaption requires some thought: Changing bin size during automatic zoom implies changing two visual properties at once, which could be visually confusing. One idea is to only change bin size by a factor of 2 during automatic zoom, since this partially preserves the geometry of the histogram bars.

3.7 Handling data storage

In a streaming data visualization application, data that is not explicitly stored is lost forever. Since data rates and volumes can be high, we might not have enough storage capacity to explicitly store every received data item. This trend will likely continue, since data volumes have historically been growing faster than available storage capacity [39]. We separate between the *data buffer*, which is all data currently available for visualization, and the *data archive* which is a separate off-line storage which can be referenced on demand. A different implementation might not distinguish between these, but it is relevant to mention the distinction in order to consider applications where data volumes are truly massive.

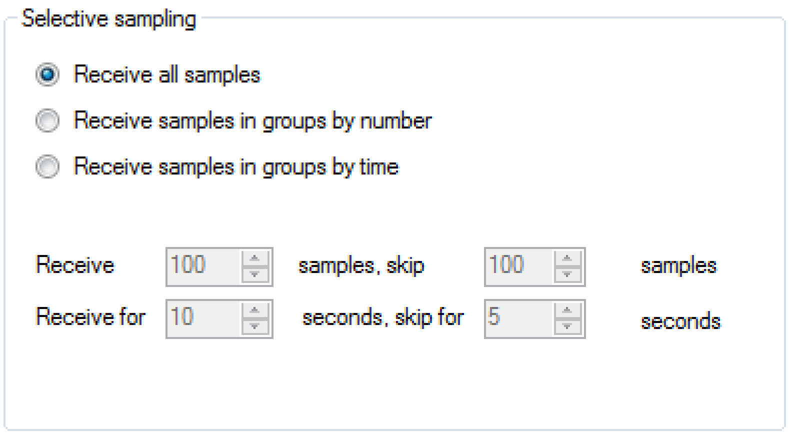
3.7.1 Filtering

The most obvious solution to a storage capacity problem is to store less data. This is sensible if there is a cyclic or intermittent aspect to our data source, such that only part of the data is of interest to the operator. This could be the case if we are e.g. monitoring processes that run only at night or every other hour. We have implemented such a capability, although its usage is limited by each particular application (see figure 3.28).

3.7.2 Compression and aggregation

Second, most data of interest to humans is non-random and has a large amount of statistical redundancy. All archived data can be compressed using a lossless compression algorithm such as LZ77 [81] or LZ78 [82]. If this is not sufficient, we need to store an aggregated form of the data.

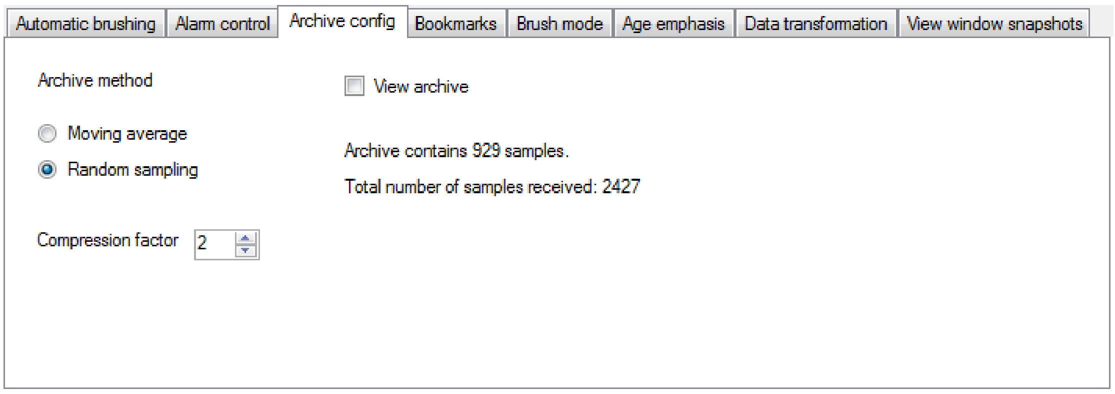
We have implemented two such schemes: Either data is aggregated by storing only the average of the last n values of each dimension, or only a *random sampling* of the last n


Selective sampling

[✓ Receive all samples]
Receive samples in groups by number
Receive samples in groups by time

Receive 100 samples, skip 100 samples
Receive for 10 seconds, skip for 5 seconds

FIGURE 3.28: Control panel for data rate reduction by intermittent sampling.



Archive method

Moving average
[✓ Random sampling]

Compression factor 2

View archive

Archive contains 929 samples.
Total number of samples received: 2427

FIGURE 3.29: Control panel for the data archive.

values is stored (control panel illustrated in figure 3.29). Both of these schemes reduce the data volume by a factor of n, which obviously leads to a loss of information. The compression factor can be further increased by lossless compression.

Our application stores all data which leaves the data buffer in a separate archive, optionally in an aggregated form as described above. This data is made available for reference at a later point in time, but is not part of the main data buffer. Such a data archive could be kept completely separate from the main data buffer. We would like the opportunity to display either the archive or the main data buffer as context when navigating these separate buffers, so we create an optional buffer visualization that displays both (see figure 3.30). This view is also compatible with the lensing technique described in 3.8.3.

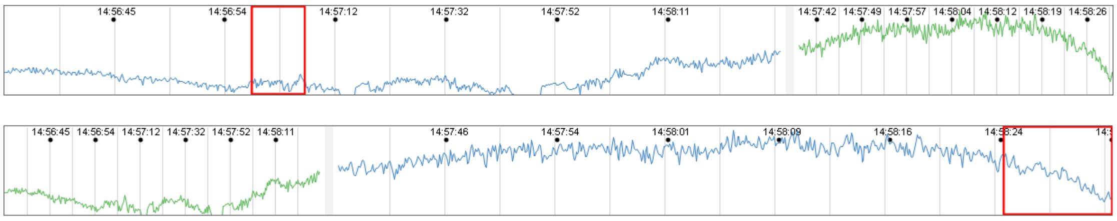


FIGURE 3.30: Buffer overview for archive and main data buffer. Archive is on the left. Top view displays archive in focus, bottom view displays main buffer in focus. This archive has a compression factor of 2, which implies that the view window will be of half size wrt. the main buffer when viewing the archived data.

3.7.3 User-directed storage

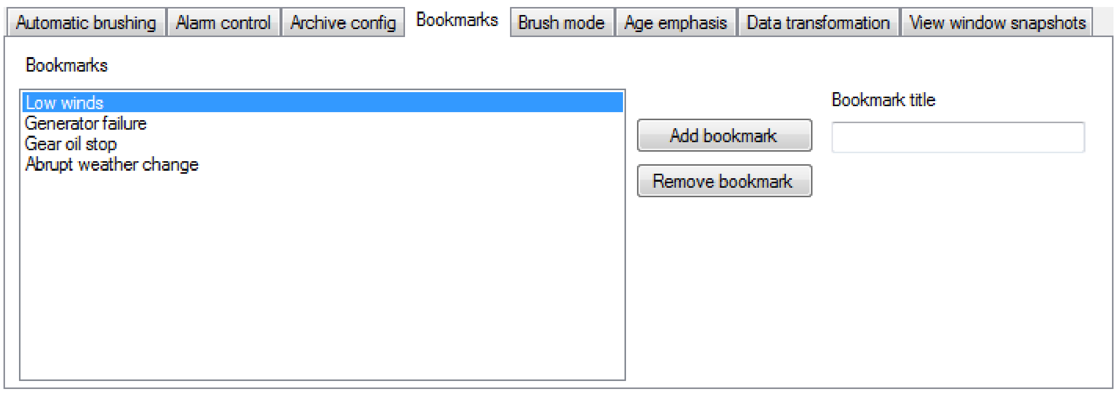
If data is archived in an aggregated manner, we will be unable to view the historical data in its original form. Therefore, it could be useful to allow the operator to store an explicit copy of the view window contents (or optionally, the entire data buffer) if a particularly interesting event occurs[17]. This is discussed in detail below, but for now we conclude that such capability is a useful way to maintain copies of important information when the storage capability is insufficient to keep up with high data volumes.

3.8 Referencing historical data

Closely related to the subject of how to *store* historical data, is how to ensure that a human operator can efficiently *find* relevant historical data for a given situation. Large data volumes can make it difficult to find data that is relevant for our current analysis scenario, even if we know exactly what we are looking for. Data volumes are unbounded, but visualization space is always limited to the area of our monitor. Also, human perceptual capability is limited even if we had infinite space for visualizing our data.

Fortunately, this is a subject which has already been thoroughly investigated in information visualization. The only challenge is to adapt these methods to a real-time application.

[17]Another option would be to create a system that performs this task automatically, by monitoring the user's behavior and storing any part of the data buffer that the user investigates closely (this could almost be used as the definition of "a particularly interesting event"). Such an approach leads to its own challenges, regarding how to organize the automatically stored data for easy retrieval. But it's worth considering, because such a function would help free the user's attention from choosing which pieces of data to keep and which to discard.



Low winds
Generator failure
Gear oil stop
Abrupt weather change

FIGURE 3.31: Control panel for storing and viewing bookmarks in the data buffer.

3.8.1 Bookmarks

The easiest way to reference older data is to know the exact location of the data we are looking for. We have implemented two such techniques: The first allows the operator to store the exact position in the data buffer that she is currently interested in (control panel illustrated in figure 3.31). This can be used to highlight relevant portions of the data buffer, and is most useful when the data buffer is large. Such bookmarks are necessarily deleted once the highlighted data leaves the data buffer, although they could also be transferred to the archive.

A second option is to do what we described earlier: The user is allowed to store an explicit copy (snapshot) of a piece of relevant data. The advantage of this technique is that it is immune to "aging"; the stored data will be available regardless of whether the data is still in the data buffer. Regardless of which technique we use, bookmarked data can either be viewed directly or in the context of the current data. By doing the latter (illustrated in figure 3.32), we are taking advantage of preattentive processing in order to help the operator see similarities between the stored data and the current situation. This could be a very useful feature, e.g. if the stored data represents dangerous conditions or conditions that forecast a particular event. By enabling this multi-dimensional pattern recognition across multiple views, we allow the human operator's cognition and domain expertise to do work that is unsuitable for the computer. This is a good example of the power of visualization.

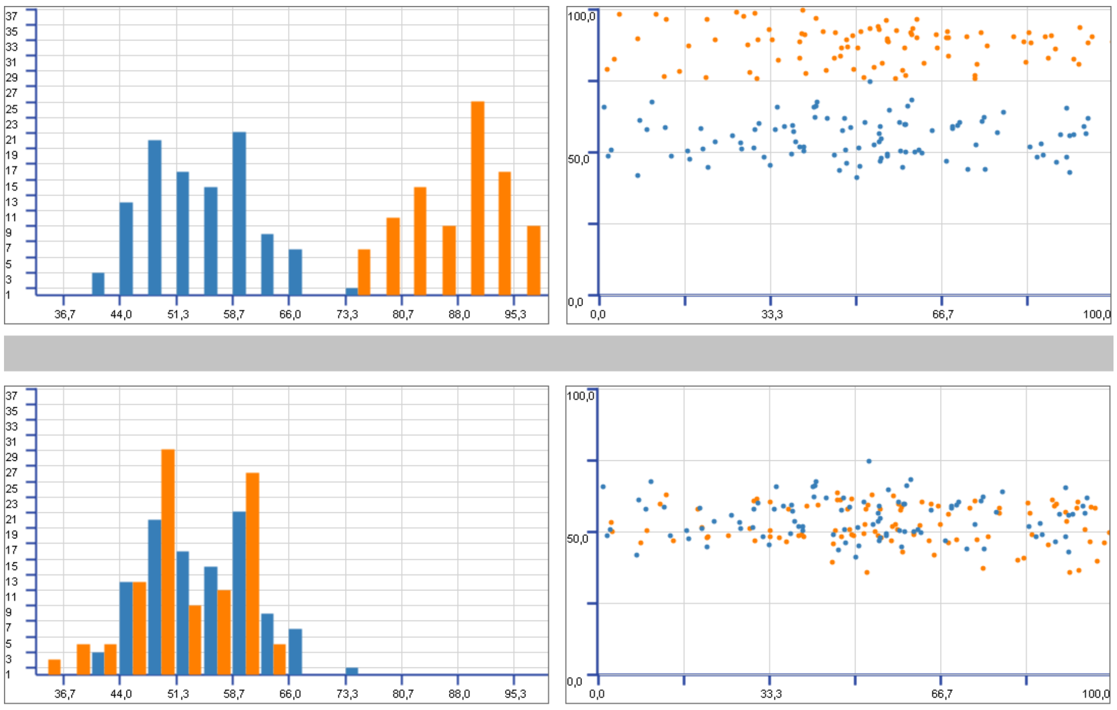


FIGURE 3.32: Two "data snapshots", displayed in the context of the current view window. Snapshot is orange, view window content is blue. Histogram bar width is halved, such that snapshot and data can be displayed side-by-side. Note the preattentive similarity between snapshot and view window in the second example. Individual data items are quite dissimilar, but the trend is evident.

3.8.2 Search strategies

If we don't have an exact reference to the data we are looking for, we need to search for it. Time series data is less structured than textual data, and we cannot expect an operator to exactly define the time series that he is looking for. Hence, search strategies for time series data must be heuristical and fuzzy. Multiple such methods exist. Wattenberg suggested a system to sketch the rough characteristics of the time series we are looking for [77], and similar techniques were further developed by Hochheiser and Shneiderman [32]. Lin et al. developed abstraction techniques that can be used both for search, pattern detection or anomaly detection [52].

We didn't want to go to the effort of replicating these techniques in the real-time domain, but note that such techniques are necessary in any streaming data analysis application that handles large data volumes. Important concerns are how to combine such techniques and how to allow multiple searches to narrow down the results, while ensuring that the real-time aspect of the analysis process is not compromised.

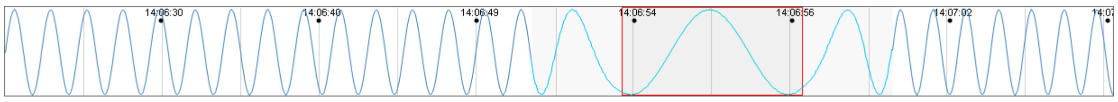


FIGURE 3.33: Lensing feature of buffer overview, illustrated on a sine curve. The view window is highlighted in red.

3.8.3 Lensing metaphors

To handle the problem of limited visualization space, we can introduce a lensing metaphor. This provides an alternative to automated search: we can visually search for a specific data feature, provided we have an idea what we are looking for and perhaps parts of the contextual data surrounding our target. In our case, this technique is especially relevant when navigating the data buffer. The data buffer can be large, and even if we know a specific feature we are looking for, it can easily be lost in the noise of the surrounding data.

We have implemented a variation of the lensing technique described by Kincaid [41]: We choose a fixed pixel width of our focus area and let the drop off area on each side have half this number of pixels. The pixel width of the lens can be selected by the user. The magnification levels in the various parts of the lens are chosen by varying the sampling interval of the data buffer, using linear interpolation where necessary. The data is sampled according to the following restrictions:

1. The contents of the view window fit completely in the focus area
2. Each drop-off area contains a number of data items equal to the view window size
3. The remaining buffer view is sampled at a constant rate, fitting all the remaining items
4. In the drop-off region, the sampling interval gradually changes between that of the focus area and the context. The change in sampling interval per pixel corresponds to the progression of an arithmetic series of length equal to the pixel count, such that the sum of interpolation intervals corresponds to the number of data items specified in (2).

These restrictions yield a lens that corresponds to a parallel projection with a linear drop-off function, as illustrated in figure 3.33.

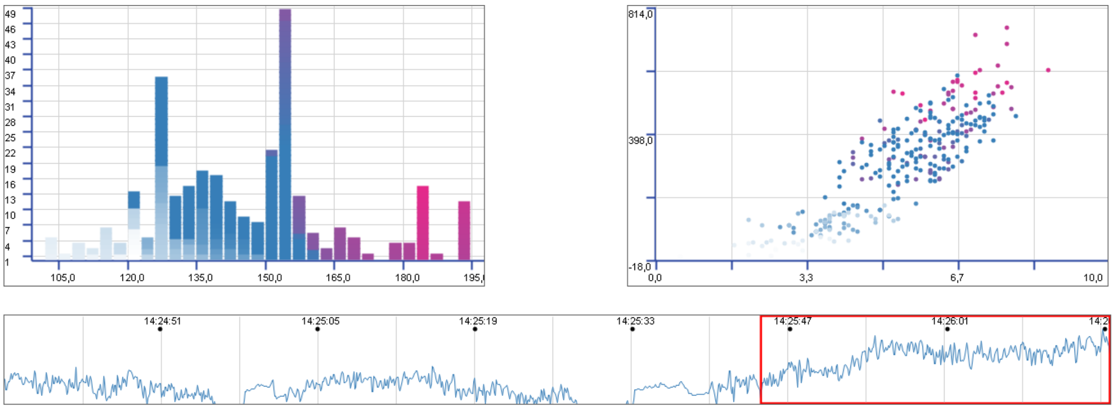


FIGURE 3.34: Illustration of age emphasis: Histogram of wind direction on the left, wind speed vs. produced power in kW on the right and wind speed in data buffer. The first and last 30% of the view window are highlighted in pink and grey, respectively. We can clearly see the trends in wind direction and wind speed, and how changes in the latter affects power production over time. The view window spans approximately five hours of data, played back at a high rate.

In retrospect, it turns out that the lensing strategy we described above is sub-optimal: The context is not fixed when we move the lens around, which could cause a degree of user confusion. Implementing a lens metaphor can be tricky, and it could be a good idea to stick with a verified approach (such as Kincaid's SignalLens [41], introduced on page 26) rather than re-inventing the wheel.

3.9 Avoiding loss of the time context

Video of age emphasis: http://vimeo.com/geirsmestad/iva-ageemphasis

Line charts of time series have the time dimension explicitly coded as position on the X axis. For other plot types, this link to the time dimension is not necessarily explicit, often only implied by the order in which items are added to the animated visualization. Hence, an operator that does not pay attention to the updates of a scatterplot or a histogram will lose track of the order in which the data arrived. This is still more informative than if the visualization was completely static, but we can do better.

The age of the data items in a histogram or a scatterplot can be visually coded e.g. by the color of each data item. A percentile of the items closest to each edge of the view window are colored by a scale corresponding to the distance from the edge. This allows us to let older data points fade out of view as they become irrelevant, and also to highlight new items such that they are brought to immediate attention (illustrated in figure 3.34). This technique allows the operator to see the time aspect at a glance, but also guides the user's attention when viewing the visualization in real time. Although the technique could be used on static data, the real-time aspect provides the ability to map the concept "now" to a visual parameter. This is a possibility that does not exist when analyzing static data.

If we don't explicitly map visual parameters to data age, the operator loses the ability to view the order in which data items arrived in the visualization. In this case, the operator must rewind or loop through the data to review the order of events, which is clearly not the optimal situation.

This concludes our discussion of how to extend IVA for streaming data, and how to handle the analysis challenges that show up in real-time scenarios. We now move on to evaluating the suitability of these techniques and demonstrating them on real data.

Chapter 4

Evaluation and demonstration

Evaluating visualization research is often challenging. The visualization process consists of many complex interactions, there is great variation in the datasets, tasks and users that techniques will be demonstrated on, and it is often hard to find a good way to measure success [19]. In our case, we are also exploring a novel area where multiple case studies in different domains are required for a thorough validation of our techniques.

We present an informal study which includes demonstrations and interviews with domain experts in the field of streaming data analysis. While we cannot prove that the techniques presented in this thesis represent the best generalized approach to streaming data IVA, this evaluation supports that our techniques have merit and deserve further investigation.

4.1 Application domain

As briefly mentioned earlier, this thesis was written in collaboration with the *Wind Operations Strategy and Support* division (from now on referred to as "Wind Operations") at Statoil ASA [6]. The Wind Operations department provided us with sensor data from the Hywind demonstration offshore wind turbine south of Karmøy, and also from the full-scale 88-turbine Sheringham Shoal wind farm north of Sheringham in the United Kingdom [4, 5]. The Hywind turbine is 65 meters tall, has a rotor diameter of 84 meters and produces 2.3 megawatts at full production. The turbines at Sheringham Shoal are 80 meters tall, have a rotor diameter of 107 meters and produce 3.6 megawatts each, at full production.

It has been our hypothesis that the sensor data from these sources represent one instance of a situation where both the *monitoring* and *analysis* aspects of data analysis are relevant. Data from Sheringham Shoal are used for the practical demonstration, and also as background for our discussions with the domain experts afterwards.

4.2 Demonstration of streaming data IVA

We first present some demonstration scenarios where we apply streaming data IVA to data from the Sheringham Shoal offshore wind farm. As we mentioned in the introduction, this data is played back faster than it would be in reality. The purpose of this is to demonstrate that our techniques are useful when there is considerable time pressure in making sense of the data. The dataset is from November 2013 at a resolution of one minute per data point, and displays data from two wind turbines. The data is played back at 7 samples per second. The following parameters are used in the demonstration: Produced power (kW), wind speed (m/s), wind direction (degrees) and turbine blade angle (degrees). Turbine blade angle (pitch) denotes the degree to which the turbine "releases" the wind in order to keep RPM, generator temperatures and produced power within design limits. Hence, low blade pitch means that the turbine produces the maximum power possible in the prevailing conditions, while high blade pitch means that the produced power is lower than the theoretical maximum. This parameter is critical to turbine operation, as it keeps the turbine operating within safe limits. With a few exceptions, turbine 1 (T1) is displayed on the left and turbine 2 (T2) on the right.

Since the demonstration scenarios are displayed in the form of annotated videos and show the entire interface of our demonstration application, an HD monitor is required in order to fully view all the details. The techniques are demonstrated in arbitrary order, emulating a real-life analysis scenario. We provide a detailed description of each scenario below; this could be used as detailed commentary to what happens in the videos.

If the Vimeo videos in this chapter are password-protected, please use the password "visgroup" (no quotes).

4.2.1 Scenario 1: Basic IVA

Demo scenario 1: http://vimeo.com/geirsmestad/iva-general

View setup

This is a scenario demonstrating the general features of streaming data IVA. The data stream is running as the clip starts. We first set up a scatterplot of T1 with wind speed on the X axis and power production on the Y axis, in the top-left view. This particular scatterplot is often called the turbine's *power curve* by wind energy professionals. The power curve generally approximates the logistic function, and the scatterplot hence forms an S-curve with a wide stem. Wind farm operators prefer this curve to be shifted as far to the left as possible, meaning that low wind speeds yield high power. The power curve of T2 is set up in the top-right view. We also set view limits to these views that correspond to the normal range of the data[1].

Investigating missing data

By brushing T1, the first observation we make is that some of the data points are going missing in the view. We pause the view window to investigate, and expand the view limits of T1. We notice that the missing data points show up with a power value of -15, which could indicate is an error condition. We resume data playback and fast-forward back to the present. We notice that production has been resumed for T1 while we were paused, and also that T2 appears to have smaller variations in wind speed than T1. We set up two time series views below the scatterplots in order to examine this difference in detail. The time series clearly show the same trend as the scatterplots: Wind turbine 1 has a greater variation in wind speeds.

[1]Setting the view limits by hand is a little cumbersome. We have previously discussed automatic zoom in detail, but this approach would not be suitable for this dataset: The data is (almost) always within a specific range, and we want to see all of this range. Automated zooming will focus on only the current data. An option for better automated setup could be to only allow the automated system to zoom out, and not back in. We had not considered this option until after we performed the demonstration.

Plateau in power production

At this point, an interesting development happens: there appears to be a plateau in power production for T1. Brushing this structure, we see that no such behavior is apparent in T2 – turbine 2 produces much greater power. A quick glance at the brushed section of the time series below indicates that this development is continuous and started at approximately 16:49:45.

We pause the view window to investigate. We bring up one new scatterplot of each turbine, this time with wind speed on the X axis and turbine blade angle on the Y axis. It becomes immediately apparent that T1 has higher pitch than T2, even though the wind speeds are similar. We interpret this as T1 operating at a higher pitch (perhaps after manual intervention) in order to limit power production to 1800kW. This makes sense, as this number is 50% of the peak production (3600kW). Later, we notice that pitch starts to increase also for T2, but T2 still has lower pitch than T1 for the same wind speed. It is apparent that T2 now operates at full output, while T1 is still at 50%. We resume playback.

We are now viewing data items in the middle of the data buffer, and have no idea what is *currently* happening to the turbines. We would like to get back to the present in order to once again monitor events in real time. Fast-forwarding, we suddenly notice that T1 is again at full power output. Brushing this structure, we see that T1 has lower blade angle for the data points where production is 100%. However, production quickly falls back to 50% capacity, and we guess that the T1 was briefly set to produce at 100% power but that the operators soon changed their minds.

Frequency-based views

We return to the present, with the view window at the head of the data buffer. Winds are moderate and power production is in the lower third. Blade angle is at a minimum, indicating that turbines are now trying to produce as much power as possible from the moderate winds. In order to illustrate a frequency-based plot type, we draw the histograms of wind direction for each of the two turbines. These plots show the same information as a meteorologist's *wind rose*, albeit drawn in a Cartesian rather than a polar coordinate system.

Brushing the wind histograms, we note that wind direction is often slightly different between the two turbines. This is interesting, as the turbines are placed quite close to each other. We note that winds are very calm during this examination, which could have implications for the inconsistent wind direction.

4.2.2 Scenario 2: Investigating quick changes in wind speed

Demo scenario 2: http://vimeo.com/geirsmestad/iva-quickchanges

This scenario demonstrates some of the more specialized techniques for streaming data IVA. We start out with the power curves as before, and draw the time series of wind speed for each turbine in the bottom plots. Almost immediately, we get quick, large changes in wind speed for both turbines. This change happened so fast that it was difficult to see what happened to the power curves during the change.

Looping and age emphasis

We select the looping tool in the buffer control window, and select the region of the data buffer where the wind speed change occurred. This section will now loop repeatedly. If we pay careful attention to the power curves, we see that the increase in power output was almost instantaneous – happening slightly earlier for T2 than T1. However, it is apparent that this order of the data points is lost if we look away from the plots for even a moment.

We therefore select the *age emphasis* tool to highlight the order in which the data items arrived. We set the selection to 0.4, meaning that the 40% newest data items will be highlighted in red and the 40% oldest will fade out to gray. Examining the power curves now, it is much easier to see the order and variability of the measurements. Notably, after the peak in power production, there is once again a sudden reduction, occuring at different times for the two turbines. This observation matches the great variation in wind speed, which can be seen in the time series below.

Keeping track of the present

The repeated looping has left us looking at older data, and we now have no idea what is going on in the present. We bring up the Extended Buffer Overview (EBO) in order to check what is occurring at the head of the data buffer. Since all the parameters of the dataset are displayed in the EBO, it can be tricky to look up the exact values of interest. This should be taken into consideration when designing the next iteration of such a system.

Regardless, we see the power output of the turbines: T1 produces 1000kW and T2 produces 1500kW. Wind speeds vary between 8-9m/s. The output and winds are fluctuating quite a lot. We fast-forward back to the present while keeping track of the EBO. After a few seconds, T1 is producing 1800kW and T2 1300kW. Apparently, wind conditions currently vary a lot.

Trend + outlier view

Since the wind speed fluctuates a lot, we bring up a trend+outlier view of T2's wind speed. We set the view to display a moving average over 5 samples, highlighting any data item that is more than 2.5 standard deviations from the mean of the view window. The plot shows that the wind is trending within a relatively narrow range, with a few outliers.

Automatic brushing

Having returned to the present, we deactivate age emphasis and prepare to use the automated brushing functionality. We initially command the system to brush all data items above the 70%-percentile of T2 power production. As an initial observation, high production in T2 does not always imply high production in T1; this can be seen by the number of brushed data items below the maximum in T1's power curve.

We again bring up scatterplots with wind speed on the X axis and turbine blade angle on the Y axis. The automatic brush for the top 30% of production on T2 is still active. An initial observation is that high production appears to correlate with low blade angle. We are curious to see if low blade angle correlates with high production, and also investigate this by setting the automatic brush to mark the bottom 40% of T2 blade angles. Looking at the power curve, this relationship appears to hold: Low blade angles correlate with high power production.

We perform the same investigation for T1 by brushing the bottom 40% of T1 blade angles. The same relationship appears to hold. This leads to an interesting question: would it have been possible to reduce the blade angle in the remaining cases, in order to increase production and hence turbine efficiency? Perhaps there are other limitations that would prevent this, but it appears to be a promising avenue of investigation.

4.2.3 Scenario 3: Archive, lensing and signal processing

Demo scenario 3: `http://vimeo.com/geirsmestad/iva-archivesandlensing`

This scenario demonstrates some of the advanced features of streaming data IVA.

Archive view

At the start of this video clip, the application has been running for a while and has received more than 4500 samples. Since the view window is only 900 samples long, most of the samples have already been dropped from the data buffer. The archive feature is set to store these dropped samples, but to compress them 50% by only storing half of them (randomly selected) and dropping the rest.

We select the "merge buffer and archive views" feature. The contents of the archive are now displayed in green to the left of the data buffer, but data buffer navigation is still allowed by moving the view window.

Lens feature

We activate the lens feature and select "View archive" in order to display the archive instead of the data buffer[2]. Playback is automatically paused, and the view window

[2]Note: The background reference lines were overlooked when implementing the lensing feature, so these are not affected by the lens. This is obviously an oversight which should be fixed in a real-world system.

size is halved to reflect the compression factor of 2. We would like to look at a greater selection of data, and double the view window size.

The spike in wind speed from the previous demonstration example is easily visible in the time series. We would like to investigate if this feature has been accurately reflected in the compressed archive. As we drag the view window towards the spike, note how the lensing feature acts as a magnifying glass on the surrounding samples.

We reach the wind spike and observe that the four views are very similar to how they looked when we first saw this event in real time. We drag the view window back and forth to illustrate how the lensing effect hides the details of the wind speed spike when out of focus, but shows all the details when in focus. The lens allows us to both locate the spike in the long time series and to see its details, something which would not (in a single time series visualization) be possible without this technique. We finally note that brushing is of course also allowed when investigating the archived data.

Basic signal processing

In order to demonstrate our signal processing features, we display the wind speed for T1 in the two bottom views[3]. We then display the moving average in the right view. Note how increasing the sample range of the moving average leads to a smoother graph, displaying trends rather than individual measurements. The moving variance can also be displayed, which shows changes in measurement dispersion rather than value[4].

We then apply the moving average to the power curves for both turbines, in the top views. Since the moving average is applied to all parameters of the dataset, it will also help highlight trends in more complex visualizations, such as scatterplots. The moving average reduces the space between individual measurements, with higher sample range leading to smaller distance. It is not obvious how this transformation induces artifacts in the scatterplots, but especially T1's scatterplot exhibits trends and clusters that were not obvious before. Using averages in scatterplots might be an avenue for further research.

[3]A minor note about the user interface in the video: View dimension settings for the X axis are disregarded for the time series views – the time series always displays the contents of the view window, regardless of X axis settings. In this example, the X dimension settings do not match between the two time series views, which due to the above has no effect for the contents of the views.

[4]Due to a programming error which was fixed after the video was uploaded, the variance in this video is erroneously calculated. Please see figure 3.17 for an accurate depiction of the moving variance.

Advanced signal processing

As described in Chapter 3, our application allows the definition of flexible linear filters. We first demonstrate this feature by applying a "box filter" to the top scatterplots, a transformation which is equivalent to the moving average. Any discretized linear filter in the form of a weighted sum with 11 coefficients or less, can be applied using this interface. We demonstrate the tent filter, which is another type of a linear smoothing filter. We also apply this filter to the time series in the bottom right view.

Finally, we demonstrate how linear filtering can be used to estimate the numeric derivative of a parameter. We first explore the data buffer to find an area where there is a marked change in the power output of T1. After locating this feature, we draw the time series for T1 power in the top right and bottom right views. Using the filter editor, we apply a *central difference* operation to the bottom right view. We change the view limits of view 4 such that the result is visible. Comparing the power graph above with its derivative below, we see that large changes in power output are also reflected as spikes in the numeric derivative. When power production falls to zero for a few moments, the derivative also goes to zero, with a spike in front and afterwards (as expected). The real-time numeric differentiation works as expected.

4.2.4 Scenario 4: Using events to visualize categorical distinctions

Demo scenario 4: http://vimeo.com/geirsmestad/iva-eventsandcomparison

This scenario demonstrates how event transformation can be used to simplify continuous parameters to high-level categories. We also demonstrate snapshots, which are used to compare different selections of parameter values.

Event definition

The example starts by opening the event manager, which lets the user define the ranges of the five event bands for each parameter. Events should ideally be used to transform parameter values into categories that have semantic meaning. We define such categories for power production and turbine blade angle. For power production, lower outlier means almost-zero production, lower and central band means low and normal production, respectively. Upper band means full (3600kW) production, while upper outlier means that production is above safe limits. For blade angle, lower outlier means erroneously low blade angle. Lower and central band means normal pitch (full utilization of the wind). Upper band means that production is restricted, and upper outlier means that production is strongly restricted. The bands of wind speeds are selected according to the statistical distribution of the wind speeds. Note that properly setting event limits requires a good understanding of the ranges and semantics of the dataset.

Event overview

After defining the event bands, we start the data stream. Event transformation automatically commences in the background. We switch to the event overview display. The event band for all parameters is displayed on a horizontal timeline, using color to denote event band. This view initially displays all the 24 parameters that are available in our dataset, but we have selected event bands only for the six first. We therefore adjust the view so only these six events are displayed. Timestamps are visible in the bottom section of the display.

After the initial twenty seconds of this data stream, blade angle for both turbines is in the central band (yellow), indicating that the turbines are attempting to produce at maximum efficiency. The lower band wind speeds cause the turbines to produce at low output. We skip ahead a few seconds, and at approximately 15:59:15, production reaches nominal levels. All parameters are in the central band, denoting normal production. A few seconds later, at 15:59:45, blade angle briefly falls into the lower band. This indicates that turbines are reducing pitch further, in an attempt to further increase production in the moderate winds. We skip ahead to 16:00:36. Production for T1 has now reached the upper band, indicating full production. Blade angle for T1 is also in the upper band, denoting that production for T1 is now being actively limited in the higher winds. The event overview is now full, and starts scrolling to the right as new data arrives.

Item-based event view

We switch to the item-based event view, which has been cleared of the recent history in order to more clearly illustrate how it evolves over time. This view displays the current event for each parameter on the left[5], with event length (in samples) coded to both a number and the height of each event bar. Event type is denoted by color and vertical positioning. Note that the X axis is not an equally-spaced timeline, although timestamps give an indication of when each event started. Each parameter has its own (non-equidistant) timeline.

If a lot of short events occur, as in this example, the item view can be distracting due to its quick and asymmetrical updates. It should work better when the distinction between bands is clearer and less frequent, but this remains to be verified. One idea is to use a trend+outlier view instead, if the data exhibits the type of large variation that would lead to many short events.

Simple frequency-based event view

It is also possible to visualize the event list as a histogram. This qualitative view displays the number of events in each category. Event length is not encoded; one long event would simply be drawn as a bar of height one. Skipping ahead a few moments, we can quickly see from the histograms that there are no upper outliers. This means that the turbines have been operating safely throughout this interval. The wind speeds are normal distributed, as expected, while blade angle has mostly been in the most efficient region (lower and central bands).

Advanced frequency-based event view

The basic histogram is limited by not displaying the length of events. We alleviate this problem in the advanced histogram. Bar width corresponds to event length, visually sorted such that the longest events are displayed at the bottom of each histogram bar. From the example, we see how the bars "grow" as new events occur. We increase the

[5]There is a technical reason why new events are inserted from the left and not from right, as explained in a footnote on page 59. Ideally, the item-based event view should have new events inserted on the right.

scale of the histograms in order to see the details more clearly: T2 power production has some long and many short events in the central band. The blade angle for T1 consists almost exclusively of short events, apart from a few longer events in the lower band. The upper band events for T1 power production has a very even distribution of event lengths. The power histograms have more (and longer) events in the upper band, indicating that this interval has significant periods of peak power production. This is verified by the T1 power curve scatterplot in the top left corner, which displays peak production in the entire current view window.

We skip to a different example, where more events are represented than before. Histogram scale is 2.00, meaning that each event gets two pixels in the vertical dimension of the histogram. Event type and length distribution is relatively even, indicating a period of varying conditions. The upper band events of blade angle for T1 are mostly short. This is reassuring; T1 has mostly attempted maximum efficiency in the moderate wind conditions.

We finish the demonstration of event transformation by pointing out that the advanced event histogram is more informative than the simple histogram, without loss of information. It should therefore be the preferred histogram view for events.

Bookmarks

We now demonstrate how bookmarks and snapshots can be used to navigate the data buffer and compare different selections of parameter values. We switch away from the event view to a regular session of streaming data IVA. Browsing the data buffer, we notice a change in wind direction in the lower right histogram. Using the bookmark feature, we label this view window selection "Wind direction change". We then locate a situation where T2 is producing less power than T1, and bookmarks this as "Turbine 2 producing lower power". The bookmarks can be used to quickly switch between these points of interest. While this particular data buffer is small enough that it can be managed manually, the bookmark feature would be useful if the data buffer was very long (e.g. two weeks). Giving a name to different features of the data would also improve the potential for collaboration between different users.

Snapshots

Snapshots are an alternative to bookmarks that allow comparative visualization. Instead of storing a location in the data buffer, snapshots store all data items in the view window. This means that stored snapshots are available even after the data buffer has been completely flushed. We browse the data buffer for interesting observations and store two: One where the wind speed is falling, and one where the winds and power production is high.

The snapshot data are displayed in orange next to the contents of the current view window. This allows for easy, preattentive comparison by looking at the selected views[6]. We fast-forward such that the view window returns to the present. Looking at the views, we see that differences or similarities between the snapshot and the current data are immediately obvious. This means that if we had a snapshot of data that represented previously known dangerous conditions (or even a known precursor to dangerous conditions), visual comparison between this snapshot and the view window would make it easy to notice when a similar situation arises. Looking at the views, we see that the present situation becomes more and more similar to the ”High winds” snapshot. The major difference is in the wind direction, which can be seen from the histogram in the lower right view. Briefly switching to the ”Gradual reduction in wind speed” snapshot, we see that this bears to resemblance to the current situation.

4.2.5 Scenario 5: Intermittent updates and automatic zoom

Demo scenario 5: `http://vimeo.com/geirsmestad/iva-zoomandintermittent`

As the final demonstration scenario, we give some more attention to the intermittent update and automatic zoom features. The Sheringham Shoal wind farm dataset is not an ideal demonstration example for these techniques: The data arrives too slowly for the intermittent updates to be of great value, and most of the data values can be reasonably expected to be within a relatively narrow range – making automated zooming largely

[6]One could also imagine a background feature that autonomously compares the snapshots with the current data values, and alerts the user if they are sufficiently similar. A simple example of such a scheme could be to treat the snapshot and the current data as two vectors (perhaps disregarding parameters that are irrelevant, selected by the user), and compare their Euclidean distance against a limit specified by the user. Such a scheme would free up the attention of the user, while still enabling a degree of manual control with the comparison.

superfluous. The motivation for this final demonstration is therefore to give a better example of how these techniques behave in practice, rather than to show that they provide us with qualitatively better capabilities.

Intermittent updates

We initially set the visualizations to update every five seconds, but perform the update almost instantly. Clearly, this simple approach makes it impossible to see the order in which the update happened. Increasing the update time to one second greatly improves perception. We also show how the view behaves when the update time is set to two seconds.

Automatic zoom

We increase the update interval to seven seconds and turn on automatic zoom[7]. View limits are now gradually adapted to display all of the visible data items. Note that the gray grid lines in the background provide context for the zooming operation. Our application allows re-setting these grid lines by clicking the "Set reference lines" button, and the reference lines are currently set to match the initial limits we selected for the views. Since this dataset has a clear and expected range for all sensor values, using automated zoom can actually be counterproductive: The absolute distribution of the current values is not immediately evident from the visualization, even though we have taken great care to perform the zooming in a controlled manner[8].

We then demonstrate why we chose the approach of zooming gradually instead of instantly: Zooming instantly when new data items arrive makes is harder to maintain a mental overview of the scale of the plots. This is especially true if the update is large, as evidenced by an update in the bottom right view. In this particular instance, abnormally

[7]Note: The manual view dimension selectors are not updated to match the new view dimensions during automatic zoom. This implementation choice was made for technical reasons, but it proved in retrospect to be confusing to the user. User interface consistency is important, so this should be fixed in a real-world implementation.

[8]This is partially inherent to the concept of large, automatic changes in the view limits. But the problem is exacerbated by the fact that we overlooked a detail with the tickmarks that show the dimensions of the view: At high zoom levels, sometimes only a single tickmark is visible on the each of the coordinate axes, which makes it impossible to verify the scale of the current view. This is another streaming data challenge where extra care is required; when zooming automatically, implementations should take care to always show at least two tickmarks for each dimension.

high blade angle (100 degrees) for turbine two has caused the view to zoom out beyond the expected range, and the instant return to regular zoom levels is visually confusing. This is a perfect example of a situation where automated zoom is helpful, because our expectation of blade angles between -6 and 25 degrees would otherwise make us miss these eleven data points that are abnormally high. But in order for the automated zoom to be useful, the zoom needs to be gradual enough that we have time to process the change. We illustrate this by increasing the update interval to seven seconds and the update time to three seconds. Watching a similar abnormal blade angle situation in the bottom right view, it is obvious that gradual zoom greatly improves perception of the abnormally high blade angles. The high blade angles are present only for a few seconds before disappearing, and the view returns to its regular zoom level.

This concludes our practical demonstration of streaming data IVA. The examples above demonstrate that streaming data IVA is helpful for getting insight into streaming data in real time. While this does not constitute a scientific proof, it is a promising result that supports the validity of our techniques.

4.3 Domain expert evaluation results

In addition to this practical demonstration, we performed an evaluation with a group of domain experts in the field of wind turbine data analysis. After presenting the methodologies described in Chapter 3 to a group of four representatives from the Wind Operations department at Statoil, we asked a prepared set of questions to investigate the validity of our approach. Their current visualization tools mainly use line charts to display developments in critical sensor values, for user-defined time spans ranging from hours to months. More complex visualization and numeric analysis is performed by exporting the data and using external tools.

Initially, we asked the question *"When analyzing data from Sheringham Shoal, how important is the time aspect during the analysis work? Could the results of the analysis be affected by changes in the data during the analysis process? Please choose on a scale from 1 to 5, where 1 is 'unimportant' and 5 is 'critical'"* To this question, a representative replied that the time aspect is "2, less important" during their analysis work. However, in the discussion that followed, the Statoil representatives said that "we need both the short-term and long-term view of the data", and on a follow-up query replied that our distinction between data analysis and data monitoring accurately reflects their division of analyst roles.

Statoil described a division between a third line, second line and a first line of operations, where the urgency of the analysis increases towards the first line. One representative described the current Wind Operations department as second line, and said that there is a separate first line department where the time aspect is more pressing and minute-to-minute decisions are made regarding wind turbine operation. The Wind Operations department generally looks at a minimum one week of sensor data at a time, and makes strategic decisions based on these observations. Their representative also noted that during rough weather events, the Wind Operations department takes a more urgent role where wind farm status is monitored on an hourly basis. This supports our belief that the operator's relationship to time and urgency can change based on external factors.

In the next phase of Statoil's wind farm development, the group is planning a closer integration between the analysis and monitoring roles, and expressed significant interest in our methodologies for combining these roles.

4.3.1 Validity of our approach for Statoil's current analysis role

Having determined that the Wind Operations department is currently operating in the less time-sensitive region of the analysis-monitoring continuum, we wanted to find out whether streaming data IVA is suitable for their current second-line analysis role. To the question *"to which degree would a visualization solution using these techniques be suitable to solve the visualization tasks* you *have, when compared to your current solution"* a representative replied "somewhat better suited". Explaining this answer in detail, he stated that streaming data IVA provides functionality that allows deeper (higher-dimensional) analysis than their current tool, [without the requirement to first export the data to external tools]. He stated that in order to be "considerably better suited" than their current tool, the IVA interface would require a number of domain-specific adaptions.

These answers are in line with our hypothesis that we have explored a number of the *general* concerns of streaming data IVA. The requirement for additional domain-specific adaptions does not undermine this claim. Further, we consider this response an indication that streaming data IVA can also be suitable for visualization scenarios that are *not* time-critical, and hence likely represents a useful generalization and extension of the concepts of IVA to the real-time, streaming data domain. This is somewhat surprising, since our initial assumption in Chapter 3 was that streaming data IVA would only be useful in situations of both high data rate *and* a requirement for making quick decisions. The indication was further strengthened by Statoil's reply to our next general evaluation question: *"Does a visualization solution like the one demonstrated solve the same challenges as your current solution, or does it solve different challenges?"* The representative replied that "it can be used to solve both the same type of challenges, and also different challenges".

While this evaluation is positive and validates that our approach can be useful, we note that the Wind Operations group are operating in the less time-sensitive region of the analysis-monitoring continuum. Therefore, the expert evaluation so far has not yet answered our main question, namely whether streaming data IVA is useful in scenarios where the time pressure is high. We return to this subject shortly.

Evaluation of individual visualization techniques

After this general discussion about the applicability of streaming data IVA techniques to Statoil's analysis tasks, we provided a questionnaire grading each of the visualization techniques on a five-point scale, from 1 (useless/insignificant) to 5 (very useful). We first report the results of the questionnaire, before discussing some of the responses in detail. The context for this questionnaire was once again Statoil's current second-line analysis tasks, where the time aspect is less important.

Time navigation and time control:

Manual selection of the time span of interest on an interactive timeline	1 2 3 4 [✓ **5**]
Navigation controls for real-time data playback ("play", "pause", "fast-forward", "rewind", "loop")	1 2 3 [✓ 4] 5

Overview of new events:

Overview over current and recent values for a large number of parameters (EBO)	1 2 [✓ 3] 4 5
User-configurable alarms, triggered if a selected parameter is outside of a specified range	1 2 3 4 [✓ 5]

Handling ”fast” data:

Use of frequency-based visualizations (e.g. histograms) rather than displaying each data point individually	1 2 3 [✓ 4] 5
Event transformation: Transforming each parameter to an interval system, where each data item is at all times denoted by its current interval (from ”lower outlier” to ”upper outlier”)	1 2 3 4 [✓ 5]
Event transformation: Overview visualization	1 2 3 [✓ 4] 5
Event transformation: Item-based visualization	1 2 [✓ 3] 4 5
Event transformation: Frequency-based visualization	1 2 3 [✓ 4] 5
Periodic updates rather than continuously animated views	1 2 [✓ 3] 4 5
Filtering techniques (moving average, moving variance, generalized linear filtering)	1 2 3 [✓ 4] 5

Adapting visualizations to streaming data:

Automatic zoom which adapts the zoom level dynamically and gradually	1 2 3 [✓ 4] 5

Maintaining the overview in large data volumes:

Separating data into ”archive” and ”current data buffer” and displaying these side by side	1 2 3 [✓ 4] 5
Manual storage of bookmarks/snapshots of selected data, and displaying these text to the current data for comparison	1 2 3 4 [✓ 5]
Use of a ”magnifying lens” to make it easier to find the relevant data in buffer overview	1 2 [✓ 3] 4 5

Maintaining overview of time aspect in plots which have no spatial time mapping:

Use of colors in plots which have no spatial time mapping, in order to clearly display the order in which the data items arrived	1 2 3 4 [✓ **5**]

The questionnaire was followed by a discussion of the responses. Statoil considered the Enhanced Buffer Overview to be less relevant for their application, since none of their analysts normally monitor the data stream continuously. This response is consistent with the purpose of this view; such a visualization will necessarily only be of interest when quick reaction to new events is critical. The same response was given regarding "periodic updates rather than continuously animated views", a concept we explored in order to slow down "fast" data streams to make them correspond to the human time constraints. Statoil also considered the "magnifying lens" to be less relevant for their use, although they did not provide detailed reasoning for this response. In a previous interview, we noted that they often navigate in their data by date and time rather than by visual features of the data, which could help explain this response[9]. While these responses express only a smaller interest in some of the techniques we have developed, they do not invalidate any of our assumptions regarding streaming data visualization.

Statoil expressed great interest in the use of configurable alarms on selected parameters. This is not implemented in their current solution, and could be used to easier detect when a particular wind turbine needs additional attention from the second-line operations crew. Given the large number of data sources of the same type (88 identical wind turbines), the team suggested that statistics and comparative visualization of each parameter could be used to further improve such an alert system, providing a solid baseline for detecting abnormalities. This could be an interesting avenue for further research. One of the team members mentioned that the wind measurements from the turbines have considerable error bars. We have not considered the possibility of handling uncertainty in the data, and note that while filtering could be one way to alleviate this problem, handling uncertain measurements in streaming data is another interesting subject for future research[10].

[9]This could be an indication that it would be worthwhile to look closer at the users' routines for browsing large data volumes. It would be a big oversight if we have not sufficiently communicated the benefit of visual interaction in all stages of the data analysis process.

[10]Handling uncertain measurements is a subject which is often overlooked in visualization research [35]. Johnson et al. have explored this subject in detail, and outlined its challenges and future direction [36].

The operations team highlighted event transformation as a promising, novel technique to simplify large data streams and quickly detect abnormal behavior. While the team expressed enthusiasm for this concept, they found our item-based event visualization to be a little confusing. The team emphasized that such visualizations should make sure that not too much data is shown at once. This is an indication that our notion of event transformation is a principally sound concept, but that additional adaptions and user studies are required in order to improve the visualizations. Multiple members of the team also expressed interest in our color-coded scatterplots and histograms, where old and new data items are represented by changes in color hue. Displaying time/age in plots that have no direct mapping between spatial dimensions and time, appears to be potentially useful, even though it has not yet been considered by the team.

4.3.2 Validity for time-critical streaming data analysis scenarios

Since Statoil's current streaming data analysis tasks are in the less urgent region of the analysis-monitoring-continuum, the evaluation has so far mostly been concerned with streaming data analysis scenarios that do not have high time pressure. Analysis scenarios with higher time pressure are central to the theme of this thesis. Therefore, we would ideally want to perform a detailed evaluation of a more time-critical scenario, in order to see whether the techniques we developed have merit also under these conditions. Getting access to such an analysis/monitoring process has been difficult, and we have only been able to approximate this scenario by performing demonstrations where the data is played back at higher speed.

Since the Wind Operations group are domain experts on the analysis of wind turbine sensor data, we found it informative to ask them whether they would consider our methods useful for a more time-critical scenario. To the question *"to which degree do you think that a visualization technique using the techniques we have presented, has the potential to contribute to the work of an analyst in a* time-critical *analysis scenario on* real-time/streaming data?", the representative replied "good to very good potential". We consider this response a good indication that streaming data IVA is a useful technique also in a urgent real-time analysis scenarios. However, we must emphasize that this domain expert opinion does not constitute a formal evaluation, and that additional investigation is required in order to get a conclusive answer.

4.4 Discussion of the evaluation results

The strongest result from our domain expert evaluation is that streaming data IVA is applicable to streaming data analysis scenarios where there is relatively *little* time pressure for making decisions. While this is not the type of scenario that has been the main focus of this thesis, it is still an interesting result: This means that streaming data IVA represents a *generalization* of IVA to the real-time, streaming data domain. We retain most of the parts of IVA that make it useful to perform slow and deliberate, high-dimensional analysis of large and complex datasets, while also having the ability to react immediately to new events. Using streaming data IVA, process monitoring dashboards emerge as a special case on the fast-paced end of the spectrum. Traditional, static IVA emerges as a special case on the slow-paced end of the spectrum.

Further, the evaluation provided an indication that streaming data IVA is a *promising* technique to analyze streaming data in time-critical scenarios. The domain experts indicate that these techniques *will probably* do a better job of providing timely insight into streaming data than the current state of the art. This is a less conclusive answer to the main question of this thesis than we would have liked, but it still is an indication that streaming data IVA is a sensible approach.

4.4.1 Distinction between analysis and monitoring

The discussions with Statoil made it clear that the industry often has a sharper distinction between the *analysis* and *monitoring* communities than we initially thought. In the light of this discovery, the combination of these domains under a single methodology actually represents a quite radical notion which departs sharply from the current state of the art. In retrospect, it is therefore not surprising that we were unable to find many good examples of data and applications on which to demonstrate our techniques.

Further exploration of the applications for streaming data IVA is required to conclusively validate the applicability of this research, particularly with regards to time-critical scenarios. Such research should be done in close collaboration with domain experts in real-time data monitoring. Researchers should keep in mind that streaming data IVA would represent a very different approach than the current established industry practice, but that combining *analysis* and *monitoring* in one methodology has the potential to provide additional insight capability to the current state of the art.

This concludes the demonstration and evaluation of the techniques presented in this thesis. The bottom line of the evaluation is that our techniques appear to have merit, and seem to provide a useful aid when trying to make sense of streaming data in real time. Both the demonstration and domain expert evaluation indicate that streaming data IVA has good potential as a real-time data analysis technique. Considerable work remains in exploring streaming data IVA of more time-critical scenarios, finding additional application domains and evaluating its suitability as compared to classical monitoring techniques. But as a first approximation, this evaluation indicates that we have a good result.

Chapter 5

Implementation details

The software prototype demonstrated in this thesis is written in C# on top of the Microsoft .NET Framework 4.5, using the Windows Forms library for all user interface components. The full project consists of some ~10,000 source lines of code distributed across three separate Windows applications. A considerable part of this code is machine-generated user interface hooks and definitions.

Experience dictates that superfluous programming effort is common in single-person development efforts. With regards to functionality that is common for many different software projects, the author has previously experienced losing time due to "re-inventing the wheel", but also to spending too much time understanding complex, external library functionality. Pragmatism has therefore been an important value throughout the practical part of this thesis; when faced with a choice, we have consistently selected the technical solution that was expected to provide the best result for the time invested. In retrospect, this turned out to be a good choice. Almost no time was spent fighting limitations of the technology, so the main portion of the effort could be directed towards solving our scientific goals.

5.1 Choice of technology

In academic circles, the use of closed-source software is often a contentious subject. For scientific library functionality which is intended to be re-used by others, this might not be the best choice. We have known from the beginning that interactivity and user interface design are important features of any IVA framework. Microsoft's development tools, mainly in the form of the Visual Studio IDE, debugger and related toolchain, are known to be both powerful, mature and time-saving in this regard[1].

5.1.1 .NET Framework

The .NET Framework is Microsoft's common name for a large amount of modern Microsoft-built library functionality which interfaces with the Windows operating system [13]. This functionality is interfaced from the C# standard library, which consists of a rich, well-documented collection of data structures and operating system APIs which span the entirety of the Windows ecosystem. Double-buffered graphical animation functionality and the TCP protocol are both interfaced through this framework.

5.1.2 Windows Forms

Windows Forms[2] is Microsoft's previous-generation API[3] for user interface design, providing a user friendly, high-level abstraction on top of the low-level Windows API.

Forms provides a very powerful GUI interface designer integrated in Visual Studio, which uses an event-based model similar to Qt to enable interactivity. The GUI designer is an invaluable tool when prototyping user interfaces, since it enables quick iterations where both the visual and interactive aspects of the application can be tested.

5.1.3 Tool chain

Visual Studio 2010 was the IDE of choice for this project. Source control was managed with an online beta subscription to Visual Studio Online, which acts as a host for Microsoft's Team Foundation Server source control system. This served nicely, although it is overkill for a one-man effort.

It is hard to emphasize strongly enough how much time and effort has been saved by using the Windows Forms GUI editor and the Visual Studio debugger. In particular,

[1]Although we have no reference for this claim, this is an impression one often receives when talking to programmers who have broad experience with different types of development tools.

[2]http://en.wikipedia.org/wiki/Windows_Forms.

[3]Windows Forms is now superseded by the Windows Presentation Foundation (WPF), which is more flexible but requires higher overhead in development effort and system resources.

the Visual Studio debugger allows very flexible real-time manipulation of program flow, including on-the-fly re-compilation during an active debugging session. A powerful object inspector allows observation and manipulation of in-memory data structures during program execution. This is invaluable when developing rich GUIs with animated views. Even though C# is a statically-typed language running on a virtual machine, this flexibility rivals that of an interpreted, dynamically-typed language such as Python.

5.1.4 External libraries

In addition to Windows Forms and the .NET Class Library, the prototype uses only one external library: Kent Boogart's KBCsv[4] library for parsing CSV files, used in the data source simulator. In addition, a small 21-line module for interpolating RGB colors[5] was borrowed from user "jason"[6] at the developer network Stack Overflow.

Apart from this, all common functionality in this software has been written by the author. This includes the mathematical interpolation functionality, statistical operations, a simple TCP protocol and the linear signal filtering. Although this is functionality which would commonly be borrowed from other libraries, the documentation and low-level control functionality exposed through .NET is so powerful that re-implementing these features was assumed to be the faster choice. This assumption proved sound. Since the scientific focus has been on interactivity and user experience rather than numerical accuracy, taking the small risk that there are minor mathematical inconsistencies in these common operations seemed worthwhile. We also avoided the risk of functionality being broken by new versions of external libraries, which is a problem that plagues many small-scale development efforts.

5.2 Data source simulation

A principal concern when researching streaming data is how the streaming data source should be represented. We decided early that decoupling the user interface from the data source yields the most realistic model without causing prohibitive complexity.

[4] `http://kbcsv.codeplex.com/`
[5] `http://stackoverflow.com/questions/1236683/color-interpolation-between-3-colors-in-net`
[6] `http://stackoverflow.com/users/45914/jason`

We therefore implemented a standalone application which connects to the user interface program across the TCP protocol, transmitting data items one line at a time over the network interface. Each individual data line consists of an array of double-precision decimal numbers, each representing the instantaneous value of one sensor value from the data source. Every time a new data item is received through the user interface program's TCP connection, an event is executed and the user interface updates its data structures and views. In a real-life implementation, the data source application could be considered part of the translation layer between sensor hardware and the operator's analysis environment.

At connection time, the data source simulator also transmits a list of strings denoting the label of each dataset dimension. This is purely for operator convenience, but it is nevertheless an essential consideration for the user experience.

There are two options for how to generate the data lines that are transmitted over the network: Either they are generated automatically through a couple of hard-coded mathematical functions, or they can be parsed and played back from a CSV file on disk. In practice, the latter option proved to be more flexible both for synthetic datasets and real sensor data. Hence, this option was used almost exclusively when testing. When using a CSV file representing real sensor data, such as that of Statoil's Sheringham Shoal offshore wind farm dataset, this data source model provides an accurate analogy for what data would be available to the visualization interface when looking at a real source of streaming data.

5.3 Software architecture

The visualization environment is distributed across two applications:

- Data source simulator for streaming data to the visualization interface
- User-configurable visualization interface

The main visualization interface is a monolithic Windows Forms application, with all functionality except the Event Transformation exposed in a single window. Controllers for advanced functionality (e.g. alarms, automated brushing, bookmarks, events) are exposed in separate classes and interfaced from the main windows. We refer to Appendix A for annotated, full-size screenshots of the full visualization interface.

5.3.1 Notes about API choice and code re-use

It's worth to note that the choice of Windows Forms as the GUI API would not have been ideal for a commercial application, due to lacking separation of user interface definitions and application logic. This is an inherent feature in Forms, and is also what makes the framework so suitable for rapid prototyping. The code base becomes hard to maintain when such an application grows large. In the end, this architectural choice tied in nicely with our goal of creating a simple research prototype.

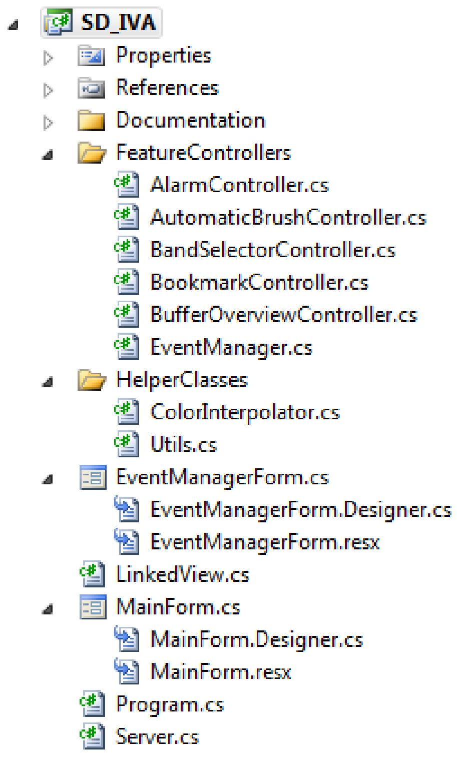
Finally, the project also re-uses (by duplication) a certain amount of common code, e.g. that of graphical buffer management. An idiomatic software engineering effort would abstract more of this behavior out to common libraries, instead of duplicating it for different parts of the program. However, this pragmatic choice turned out to be a time-saving measure, as minor differences in the behavior of common functionality (e.g. between the histogram and moving-average views) were now trivial to implement and did not require maintaining a long, separate list of special cases.

5.3.2 Data model

The data buffer described in Chapter 3 is central to all operations performed by the visualization interface. While we have described this data structure as a FIFO queue in terms of data flow, random access is essential for many of the required operations.

We decided to use an array list to store this data buffer. This implies an O(n) update whenever new data is received[7], but provides O(1) random access. Since drawing operations are the most computationally intensive part of this application and data updates are comparatively rare at e.g. 10 updates per second, this choice proved unproblematic. For a more scalable program that handles greater data volumes, a different data structure should be considered. One possibility would be a hash map with sequentially

[7]O(1) insertion of the new data line, followed by O(n) deletion of the data line which leaves the buffer.



```
◢ SD_IVA
  ▷ Properties
  ▷ References
  ▷ Documentation
  ◢ FeatureControllers
      AlarmController.cs
      AutomaticBrushController.cs
      BandSelectorController.cs
      BookmarkController.cs
      BufferOverviewController.cs
      EventManager.cs
  ◢ HelperClasses
      ColorInterpolator.cs
      Utils.cs
  ◢ EventManagerForm.cs
      EventManagerForm.Designer.cs
      EventManagerForm.resx
    LinkedView.cs
  ◢ MainForm.cs
      MainForm.Designer.cs
      MainForm.resx
    Program.cs
    Server.cs
```

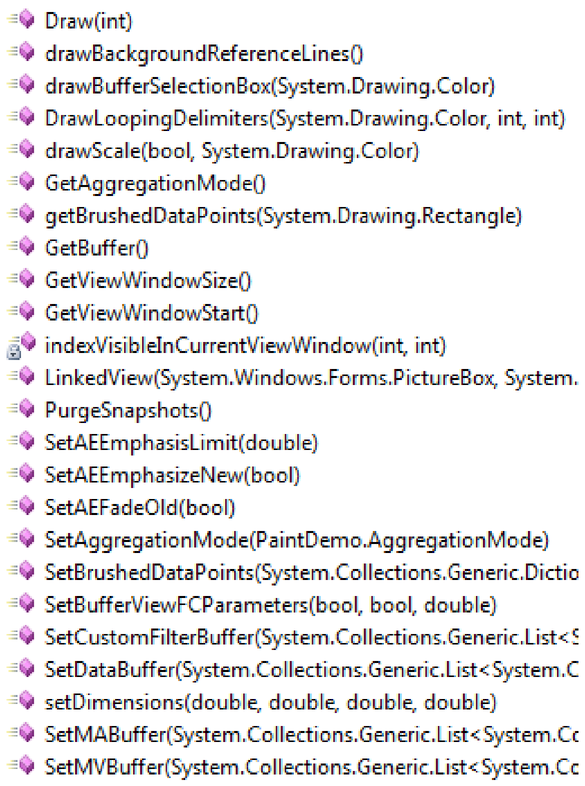
FIGURE 5.1: Class overview for the streaming data IVA application. The visual layout of the main user interface is specified in *MainForm.Designer.cs*, and the user interface behavior is specified in *MainForm.cs*. The classes in the *FeatureControllers* folder specify behavior for advanced features, and these classes are instantiated from *MainForm.cs*.

increasing keys, but a performance study would clearly be required in order to avoid premature optimization.

Individual data items that are stored in this list, follow the same convention as described in the data source simulator: Each data item is a list of double-precision numbers, each number representing the value of a single dimension of the dataset. Using double-precision numbers also allows us to represent categorical data, where each category is denoted by a specific integer number. The Sheringham Shoal dataset also uses this convention for some parameters, e.g. the current status code of each wind turbine.

5.3.3 Details about the code structure

Windows Forms applications commonly gather all user interface behavior for a single window in one large class. Hence, our *MainForm.cs* class contains a large amount of user interface code and is the controller for the main visualization application. This class also handles all user interface events (mouse clicks, button presses) and forwards them



```
Draw(int)
drawBackgroundReferenceLines()
drawBufferSelectionBox(System.Drawing.Color)
DrawLoopingDelimiters(System.Drawing.Color, int, int)
drawScale(bool, System.Drawing.Color)
GetAggregationMode()
getBrushedDataPoints(System.Drawing.Rectangle)
GetBuffer()
GetViewWindowSize()
GetViewWindowStart()
indexVisibleInCurrentViewWindow(int, int)
LinkedView(System.Windows.Forms.PictureBox, System.
PurgeSnapshots()
SetAEEmphasisLimit(double)
SetAEEmphasizeNew(bool)
SetAEFadeOld(bool)
SetAggregationMode(PaintDemo.AggregationMode)
SetBrushedDataPoints(System.Collections.Generic.Dictio
SetBufferViewFCParameters(bool, bool, double)
SetCustomFilterBuffer(System.Collections.Generic.List<S
SetDataBuffer(System.Collections.Generic.List<System.C
setDimensions(double, double, double, double)
SetMABuffer(System.Collections.Generic.List<System.Cc
SetMVBuffer(System.Collections.Generic.List<System.Cc
```

FIGURE 5.2: Excerpt of class structure for *LinkedView.cs*. This is a big class with a large number of methods and instance variables, so the full list of methods and variables is not included[8]. For examples of a few of the methods, *Draw* commands the view to redraw its contents, *getBrushedDataPoints* commands the view to return all data points encompassed by the current brush and *setDimensions* commands the view to change its data space dimensions. All these methods are public; the convention of letting capitalization denote protection level has not been followed.

to the appropriate controller. The class overview for the streaming data IVA application is illustrated in figure 5.1.

The behavior of the linked views, which is key to the application, is specified in *Linked-View.cs* (excerpt provided in figure 5.2). This class specifies the behavior for all view types, and have public methods that are called to select the view type that should currently be displayed. View limits and other view behavior is also specified in this class.

[8] A deliberate choice was made to keep functionality for all views in a single class, instead of subclassing from a common view class with shared functionality for each plot type. We note that this violates the textbook of software development, but defend this choice as reducing the development overhead when the nature of the future structural changes is uncertain. This does introduce the requirement that each developer have a good idea of the structure of the class, but this is not an issue in a one-man development effort. This decision would have caused problems if this software prototype had to be made considerably bigger, but it proved to be a sound decision under the prevailing constraints. Prototyping is different from writing production-quality software, with its own benefits and drawbacks.

Five instances of *LinkedView.cs* are created during program startup, after which each is connected to one of the four linked views (and one to the data buffer overview) in the main visualization interface. At this point, each linked view (and the buffer overview) has a reference to the main data buffer described under *Data model* (5.3.2), which resides in *MainForm.cs*. After a series of other steps performed during the initialization of *MainForm.cs*, the application can now connect with the data source simulator. This is performed by the main window holding an instance of *Server.cs*, which accepts a TCP connection from the data source simulator. Once this connection has been established, all data items concurrently received from the data source over the TCP connection will be stored in a thread-safe intermediary buffer in *MainForm.cs*.

After the TCP connection has been established, a timer in *MainForm.cs* will execute the main loop of the application every 40 milliseconds (25 times per second), or as fast as resource constraints permit. The main loop is as follows:

1. Move any newly received data items from the intermediary buffer to the main data buffer. If the buffer is full, purge any overflowing data items.
2. If a brush is specified in any of the views: Command this view to return the indexes of all data items encompassed by the brush, taking the selected brushing mode into account.
3. Based on the result of this operation, tell all the views which data items are now brushed.
4. Update the position of the view window, depending on the playback mode and whether any new data items have been received since the last iteration of the main loop.
5. Command all views to redraw their contents, taking into account the brush contents specified in (3).

This enumeration is obviously a simplification. *MainForm.cs* also holds instances to all the *FeatureControllers* classes, and many of these classes are also updated during the main loop. These classes contain behavior for alarms, the Enhanced Buffer Overview, automated brushing and bookmarks. This summary is only intended to give a high-level overview of our application; source code is available on demand if there are any more specific questions.

Note that event transformation and visualization is handled by *EventManagerForm.cs*, *EventManager.cs* and *BandSelectorController.cs*. This functionality is compartmentalized, and shares little behavior with the rest of the application apart from the connection to the main data buffer. The *HelperClasses* folder contains various functionality, including color interpolation, linear filter calculation, linear interploation of data values and statistical functions.

5.4 Color mapping

In any visualization system, choosing a good color scheme is an important task. This is a tradeoff between perceptual accuracy, acceptable contrast and making sure the interface is kind on the eyes. In addition, ensuring acceptable usability for colorblind users is also a concern. Harrower and Brewer's excellent ColorBrewer web application [26] enables a good balance of these concerns, and has been used as an aid in generating the color schemes used throughout this thesis.

Chapter 6

Summary

In the following, we briefly summarize the scope, contents and the results of our work. For additional references to related work, we refer to the main text of the thesis.

6.1 Problem domain

The visualization literature contains a large amount of work on the visual analysis of data where time is a critical parameter. However, comparatively little attention has been given to visualization scenarios where the data arrives in parallel with the user's analysis procedure. Such *streaming* data sources are frequently monitored in real time through *monitoring dashboards*, but these usually give no ability to investigate higher-dimensional properties and relationships in the data. Multiple methodologies exist for creating dynamic and interactive visualizations of complex datasets, and using these visualizations to explore complex, higher-dimensional properties in the data. However, these tools rarely give the ability to explore streaming data in real time. We are left with a distinction between real-time data *monitoring* where data is observed in real time with no ability to perform a detailed, in-depth analysis, and data *analysis*, where visual techniques are used to explore static datasets in detail. There is a lack of techniques which combine these two scenarios.

Interactive Visual Analysis (IVA) has proven to be a powerful set of methodologies for visually exploring and analyzing complex datasets, by allowing human users to apply reasoning and domain knowledge through an interactive process where complex visualizations are generated and investigated on demand. IVA has been successfully used for data exploration and hypothesis generation in a large number of fields, including engineering [59], medicine [3], climatology [37] and biology [75].

6.2 Contributions

We define a methodological framework for extending Interactive Visual Analysis such that it can be applied in real time to streaming data. The core of this framework is a *data buffer* of the available data, which is dynamically updated in real time. We select a subset of this buffer as the *view window*. The contents of the view window are made available to a number of Coordinated Multiple Views, where the user can dynamically select visualization types and brush data items to investigate higher-dimensional properties of the data in real time.

We explore many of the general challenges of real-time visual data analysis for the purposes of making decisions in real time, and investigate techniques to help solve these challenges:

6.2.1 Time management

The manual selection of the view window, looping and a general "playback metaphor" are used to allow the user to navigate in time. These techniques allow for flexible, manual navigation in a large data set in real time and, crucially, also for monitoring the data immediately as it becomes available. Previous events can be replayed and investigated closer, on demand. We discuss the *conflict of attention*, which is one of the challenges that must be handled when such time navigation is allowed: If the user is investigating previous events, the presently arriving data is unattended. This could cause the user to miss relevant observations at the time they would be most useful to make.

6.2.2 Linked views and statistics

The concept of *linking and brushing* is explored in the context of streaming data, and we discover that this concept becomes ambiguous when the data source is dynamic.

Since the contents of the linked views can change, selecting (brushing) part of the views can in fact refer to many different selection operations. Two examples of such brushing operations are "all data points that show up in a specific region of the view", and "only the specific data items that are visible at the time of brushing". We describe four such brushing operations, and also illustrate a technique for automatic brushing based on the statistical properties of the data.

A similar observation is made regarding the calculation of the statistical properties of streaming data. The statistical properties of a static dataset (e.g. mean, variance, percentiles) are usually fixed since the dataset does not change. This is not true of streaming data; such statistical measures vary both according to what data are currently available and which subset of the data the user is currently interested in. We describe some of the different subsets of the streaming data that can be useful for different purposes when calculating the statistics of streaming data.

6.2.3 Maintaining overview of the present

We investigate techniques to allow the user to maintain an overview of the present situation in the cases where the user is analyzing events that have occurred in the recent past. This is important due to the *conflict of attention*, briefly described above. We investigate three different techniques for maintaining an overview of the present: Alarms that alert the user when data values move outside of specified boundaries, overview displays that show an abbreviated form of the data currently available, and dynamic calculation of statistics on newly arriving data.

6.2.4 Maintaining overview over high data volumes

When handling high data volumes in real time, a user can easily be overwhelmed due to the limitations of human perception. We refer to the literature on perception in order to quantify these limits, which are referred to as the *human time constraints.* We then suggest a number of ideas for alleviating this problem. Data rates can be reduced through aggregation or random sampling of the data stream. Frequency-based visualizations can be used to hide the details of the individual data items that arrive. We define a *trend+outlier* view that performs smoothing on noisy data while still displaying outlier data points, allowing the user to both follow trends and to monitor deviations from the trend.

We explore various transformations that can be applied to streaming data, for the purpose of improving perception or investigating derived forms of the data. A technique for discretized linear filtering is introduced, based on the *convolution* technique as applied in signal processing. We describe how views can be updated intermittently instead of being animated continuously, which can make a noisy, high-volume source of streaming data conform to the human time constraints, potentially improving the ability of a human user to see how the data values develop over time. We finally describe a simple scheme for event transformation, which transforms a continuous parameter value into five bins, allowing the user to maintain a high-level overview of many parameters at the same time. We then explore various techniques to visually represent these events.

6.2.5 Mental map preservation

Most visualization types are not adapted to displaying a streaming data source in an animated view. When viewing a streaming data source in an animated plot, new data could arrive that are outside of the current limits of the views. We suggest automatic zooming as one technique to handle this challenge. Automatic zooming has the potential to invalidate the user's *mental map* of the relationship between the data values and what is visible in the visualization. We therefore introduce a new technique to automatically zoom in scatterplots and histograms, which reduces the probability of invalidating the user's mental map of the data.

Scatterplots turn out to work comparably well for visualizing streaming data. But when adapting histograms to streaming data, we discover some challenges due to the fact that streaming data causes the distribution of the data to change over time. These challenges are described, and we suggest one method of drawing animated histograms in a way that is less confusing for the user.

6.2.6 Storing and navigating high data volumes

A streaming data analysis scenario implies that we could receive a large amount of data which must be handled and stored if we are going to investigate it. We explore some ways to handle data storage in the case where the data buffer is too small to contain all the data received from the streaming data source. We also explore some techniques to store this data for later reference.

We describe a method for visualizing *snapshots* of historical data in order to to allow easy comparison between previous data and the current situation. We also discuss methods of finding and referring to historical data when this is required during an analysis scenario: Specifically, we investigate the use of bookmarks and lensing metaphors for visually searching in line charts. Automated search strategies are briefly discussed, but not investigated in detail.

6.2.7 Displaying time in plots without a temporal mapping

Since many plot types have no spatial mapping to time, a user which monitors an animated plot can easily lose track of the order in which the data items arrived. We discuss one technique to visually code the age of data in plot types that have no spatial mapping to time. Using scatterplots and histograms, we demonstrate this technique on wind turbine data. This technique preserves the user's mental map if he or she is unable to continuously monitor an animated visualization of this type.

6.3 Evaluation and demonstration

Finally, we successfully demonstrate most of the methods we have developed on sensor data from two wind turbines in the Sheringham Shoal offshore wind farm. This demonstration does not constitute a scientific *proof* that our techniques are sound, but it strengthens the claim that streaming data IVA is a promising technique for analyzing streaming data in situations where the time pressure is high.

After the demonstration, we present our techniques to a team of domain experts in the field of wind turbine data analysis. The domain expert evaluation shows that the techniques we have developed, are likely useful for analyzing streaming data in situations where the time pressure for making decisions is generally low. The evaluation indicates that the techniques also have good potential to contribute to the work of an analyst in a time-critical streaming data analysis scenario, although further evaluation is required in order to conclusively verify this claim.

Chapter 7

Conclusions

Exploring streaming data IVA and real-time data visualization has been a very interesting and worthwhile experience. This final chapter summarizes the major lessons learned during this project, and suggests directions for further research on streaming data visualization.

7.1 Findings

We now present the major findings of this thesis.

7.1.1 Urgency causes difficulties

For any data analysis task where humans are involved, placing a pressing deadline for the analysis introduces many additional complexities compared to analysis tasks where the time pressure is lower. In IVA, views must be selected and configured, brushing operations performed, and the user must understand what he or she sees in order to ask additional questions. All of this takes time, which means that new events can happen during the analysis. Also, the user needs to be able to finish the analysis quickly enough to take action or make recommendations based on the results. These challenges can be alleviated with the techniques we have presented. They can never be solved entirely, since the progress of time cannot be stopped and the attention of the user cannot be duplicated.

7.1.2 Most plot types are not adapted to animation

We have only used three of the most common plot types: Line charts, scatterplots and histograms. Most common plot types are *not* easily adapted to display streaming data by real-time animation. We have seen this evidenced both in the adaptions required to adapt histogram bin selection to streaming data, and to dynamically adapt the zoom levels of the three plot types in response to changes in the range of the data to be displayed. Other well-known visualization types are likely to exhibit additional problems, especially since most plots are more complex than the ones we have investigated. Researchers and developers need to keep this in mind when designing visualization tools for streaming data.

7.1.3 Statistics and brushing become more complex

Calculating averages, variance, standard deviation or other statistical measures is well-defined on static data: There is only a single dataset to refer to. When working with streaming data, such operations could be performed on many relevant subsets of the data: For example, all data which has ever been received, all the data currently available to the user or all the data which is currently drawn in the plots on screen. Each of these operations is useful for different tasks, and hence this is another complexity introduced by streaming data which is not present on static data.

Similarly, the common linking and brushing operation becomes more complex on streaming data. There is no bound to the brushing operations that can be defined, but data space brushing, view space brushing, data item brushing and statistical brushing are all obvious candidates for relevant brushing techniques. This is another complexity of streaming data IVA which researchers and developers should keep in mind.

7.1.4 Streaming data IVA is suitable when the time pressure is low

Our domain expert evaluation determined that techniques from streaming data IVA could be somewhat better suited to data analysis than their existing tools, even in the situation where the time pressure for drawing conclusions and making decisions is generally low. This is a useful result which shows that streaming data IVA can sometimes be used as a drop-in replacement for existing tools, even when the time pressure for making decisions is low.

7.1.5 Streaming data IVA is a generalization of IVA

Following the point above, streaming data IVA represents a *generalization* of Interactive Visual Analysis in the time domain. Process monitoring dashboards emerge as a special case on one end of the urgency continuum, while classical, static IVA emerges as a special case on the other end. The generalization of scientific results is usually considered good – however, there is the caveat that combining these cases in a single tool will create more complexities than each tool by itself has. We have not been able to determine with certainty whether streaming data IVA is *better* suited to streaming data analysis than each of these techniques by itself, but we have made a good case that it a useful technique and a promising avenue for future research.

7.1.6 Streaming data visualization is a complex subject

The findings above lead to the conclusion that streaming data visualization is a broad and challenging subject. We have only investigated Interactive Visual Analysis, which covers just a portion of the visualization landscape. It seems reasonable to assume that the large scope and complexities of this subject are part of the reason why relatively little research currently exists in this area.

7.2 Discussion of the domain expert evaluation results

The domain expert evaluation in Chapter 4 verified that most of our initial assumptions throughout this thesis, proved sound. Everything in this evaluation supports our notion of a continuum between data analysis and data monitoring, and indicates that there is a scarcely populated space here for visualization tools to fill. We believe that our work represents a good first iteration of generalizing IVA to the real-time, streaming data domain, providing techniques that can be used for both analysis and monitoring. Statoil's comments, both in general, regarding operational alertness during rough weather events and regarding their wish for alarm functionality, indicate that we have verified the soundness of our approach for at least part of the streaming data analysis-monitoring continuum.

Since traditional IVA emerges as a special case of streaming data IVA, the methods we have investigated can likely incorporate strategic, long-term analysis and urgent, reactive real-time analysis in the same framework. These two domains already use the same source data, so such an approach represents a good opportunity for simplification and generalization. This has the potential to simplify the training of operators, as well as providing an opportunity for closer collaboration between different layers of the organization. All else being equal, unified interfaces and interaction methods is a form of simplification which could improve the communication between different groups of technical personnel.

Monitoring dashboards represent a common contemporary solution to streaming data visualization. These systems focus on *immediate* action and discovering problems as they occur. More detailed data analysis is not given priority. Questioning this sharp distinction could be a promising avenue of research. Performing detailed analysis of complex processes in real time has the potential for early detection of problems or increases in efficiency, which is a clear indication that this subject deserves more attention.

7.3 Future work

As mentioned above, streaming data IVA is a large and complex subject, of which we have only been able to cover parts in detail. There is a lot of room for future research in this area, both regarding streaming data visualization in general and streaming data IVA in particular.

7.3.1 Discovering other applications for streaming data IVA

We were only able to find a single example use case for streaming data IVA – namely, wind farm sensor analysis. A large number of other applications are likely to exist. Effort should be spent on identifying these applications and talking to domain experts to determine whether and how IVA can be used to solve their challenges. During this work, it is important to keep in mind that streaming data IVA is a somewhat radical idea which breaks quite sharply with the current state of the art. The distinction between analysis and monitoring appears to be quite entrenched, and the idea that detailed visual analysis can be performed in time-critical scenarios will probably be met with some surprise. We have indications that there is an opportunity here for improved situational awareness and early detection of problems or inefficiencies, but this needs to be investigated in detail.

An especially promising domain for streaming data IVA are time-critical scenarios where a *detailed analysis* is *required.* Given that few options exist today for analyzing such data in a timely manner, it might be challenging to discover such scenarios where the required sensors and data collection capabilities are already in place. Where such scenarios exist, it is likely that they are currently solved by other methods: E.g. expert systems or artificial intelligence, process monitoring displays combined with very experienced operators, or other approaches where explicit, higher-dimensional analysis by a human does not play the central role. Given the human abilities in pattern recognition and abstract reasoning, there is potential for greater efficiency in such tasks, or even developing capabilities that do not yet exist. Some domains to explore could be mechanized manufacturing, chemical production or other industrial processes, space rocket launches or medical diagnosis and treatment, but it is required to examine each domain and its data sources in detail.

7.3.2 Additional evaluation of time-critical, practical applications

We have developed techniques for streaming data IVA, but we have only been able to determine its suitability for the less time-sensitive portion of the analysis-monitoring continuum. We have indications that it is also suitable for time-sensitive analysis tasks, but future research should focus on verifying this claim with practical examples – preferably by investigating if streaming data IVA does a better job than traditional monitoring tools at detecting subtle and complex patterns in streaming data. One starting point for such investigation is to talk to monitoring personnel who have to make timely decisions based on streaming data, figure out how they cope with these challenges today and see whether their tasks would benefit from such an approach.

7.3.3 Investigate different plot types and data sources

We have investigated how two-dimensional scatterplots, line charts and histograms behave when animated to display streaming data. How do other plot types work, and what are their challenges? How do parallel coordinates [33], 3D scatterplots, volume rendering or bar/pie charts work for streaming data? How can we handle different data sources and types, e.g. streaming data from volumetric LIDAR[1]? There are many unexplored questions in this area.

7.3.4 Investigate advanced IVA techniques

We only gave a brief consideration to the advanced aspects of IVA. Compound brushing, gradual degree-of-interest brushing, additional custom brush types and real-time attribute derivation have not been considered in detail. Perhaps there are other advanced techniques that work well with streaming data?

7.3.5 Exploring streaming data visualization in general

Interactive Visual Analysis is by no means the only visualization methodology that can be applied to streaming data. As we have seen, the literature is sparse on this subject. How can novel visualization techniques be applied to static monitoring dashboards? The scope of streaming data visualization widens as new sensor types become available. What new streaming data visualization techniques could be developed to handle these new sensors, and how can such techniques increase the number of data monitoring situations where a human can be in the loop and apply intuition and domain knowledge?

7.4 Closing statement

This project ended up with a larger scope than I initially anticipated, and I am pleasantly surprised that our humble Master thesis project was able to cover an area where relatively little research has already been done. Streaming data visualization is a large

[1]`http://en.wikipedia.org/wiki/Lidar`

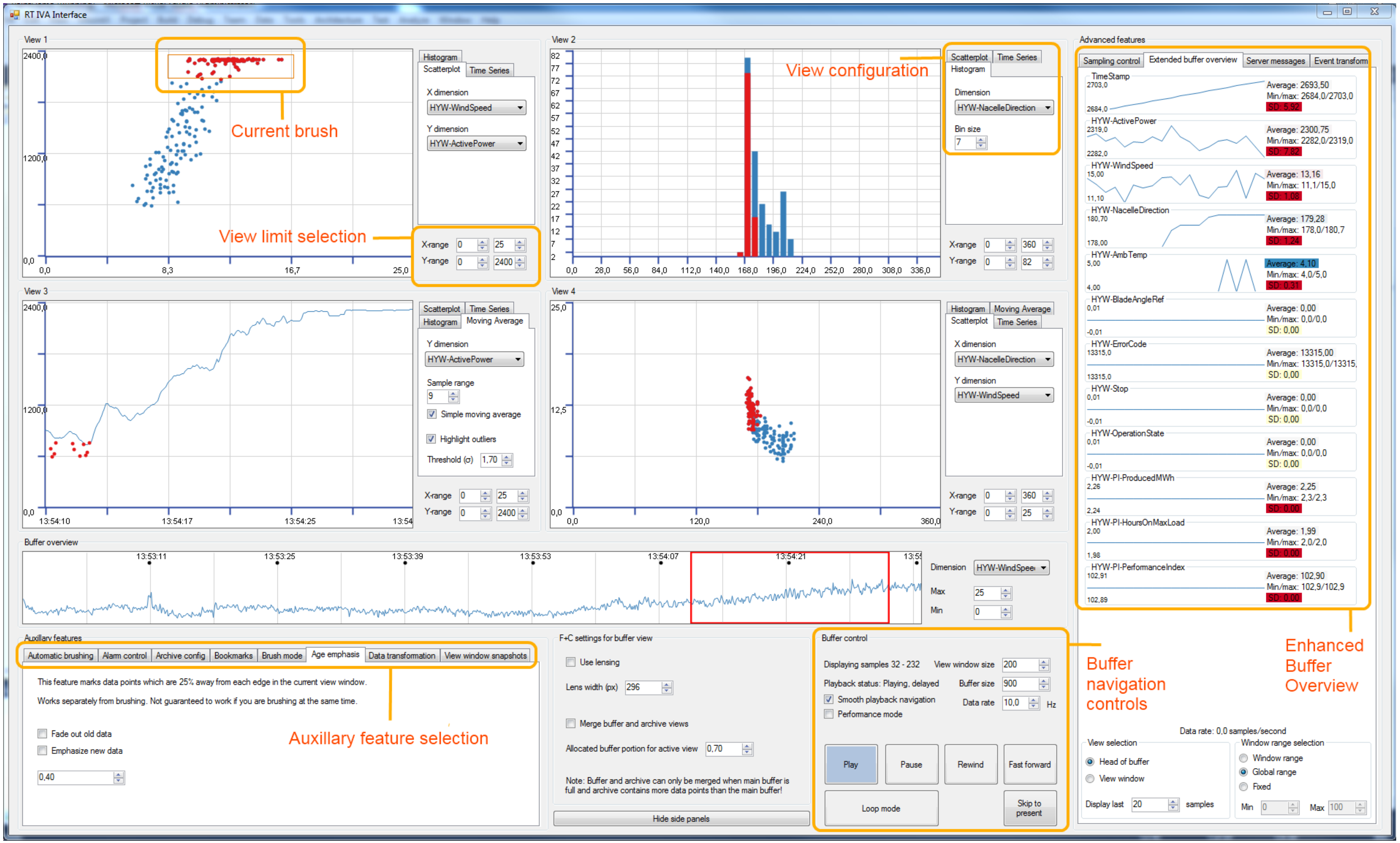
and heterogeneous subject, and it has been very fascinating to be able to cover it with some degree of generality.

After talking to industry personnel who are outside of academia, it is obvious that visual methods for data analysis can be intuitive enough to spark the interest of professionals with very diverse backgrounds. This observation should not be underestimated. Communication is vital when faced with a large and complex problem. Any tool that can help people with different backgrounds to understand each other, should be given serious consideration. It is clear that there is a great potential for solving practical problems with visualization, regardless of whether the data source is streaming or static. And for the problem of fast analysis of streaming data sources, there appears to be an open field for potential solutions. Hopefully, some of the deliberations in this thesis could prove helpful in solving such problems.

Appendix A

Full-size screenshots of IVA interface

This appendix contains screenshots of the streaming data IVA interface presented in the thesis. Source code is available on demand from geir.smestad@gmail.com.



X dimension: HYW-WindSpeed
Y dimension: HYW-ActivePower
X-range 0 25
Y-range 0 2400

Dimension: HYW-NacelleDirection
Bin size 7
X-range 0 360
Y-range 0 82

Y dimension: HYW-ActivePower
Sample range 9
☑ Simple moving average
☑ Highlight outliers
Threshold (σ) 1,70
X-range 0 25
Y-range 0 2400

X dimension: HYW-NacelleDirection
Y dimension: HYW-WindSpeed
X-range 0 360
Y-range 0 25

Dimension: HYW-WindSpee
Max 25
Min 0

TimeStamp: Average: 2693,50 Min/max: 2684,0/2703,0 SD: 5,92
HYW-ActivePower: Average: 2300,75 Min/max: 2282,0/2319,0 SD: 7,82
HYW-WindSpeed: Average: 13,16 Min/max: 11,1/15,0 SD: 1,08
HYW-NacelleDirection: Average: 179,28 Min/max: 178,0/180,7 SD: 1,24
HYW-AmbTemp: Average: 4,10 Min/max: 4,0/5,0 SD: 0,31
HYW-BladeAngleRef: Average: 0,00 Min/max: 0,0/0,0 SD: 0,00
HYW-ErrorCode: Average: 13315,00 Min/max: 13315,0/13315, SD: 0,00
HYW-Stop: Average: 0,00 Min/max: 0,0/0,0 SD: 0,00
HYW-OperationState: Average: 0,00 Min/max: 0,0/0,0 SD: 0,00
HYW-PI-ProducedMWh: Average: 2,25 Min/max: 2,3/2,3 SD: 0,00
HYW-PI-HoursOnMaxLoad: Average: 1,99 Min/max: 2,0/2,0 SD: 0,00
HYW-PI-PerformanceIndex: Average: 102,90 Min/max: 102,9/102,9 SD: 0,00

This feature marks data points which are 25% away from each edge in the current view window.
Works separately from brushing. Not guaranteed to work if you are brushing at the same time.

☐ Fade out old data
☐ Emphasize new data
0,40

☐ Use lensing
Lens width (px) 296
☐ Merge buffer and archive views
Allocated buffer portion for active view 0,70

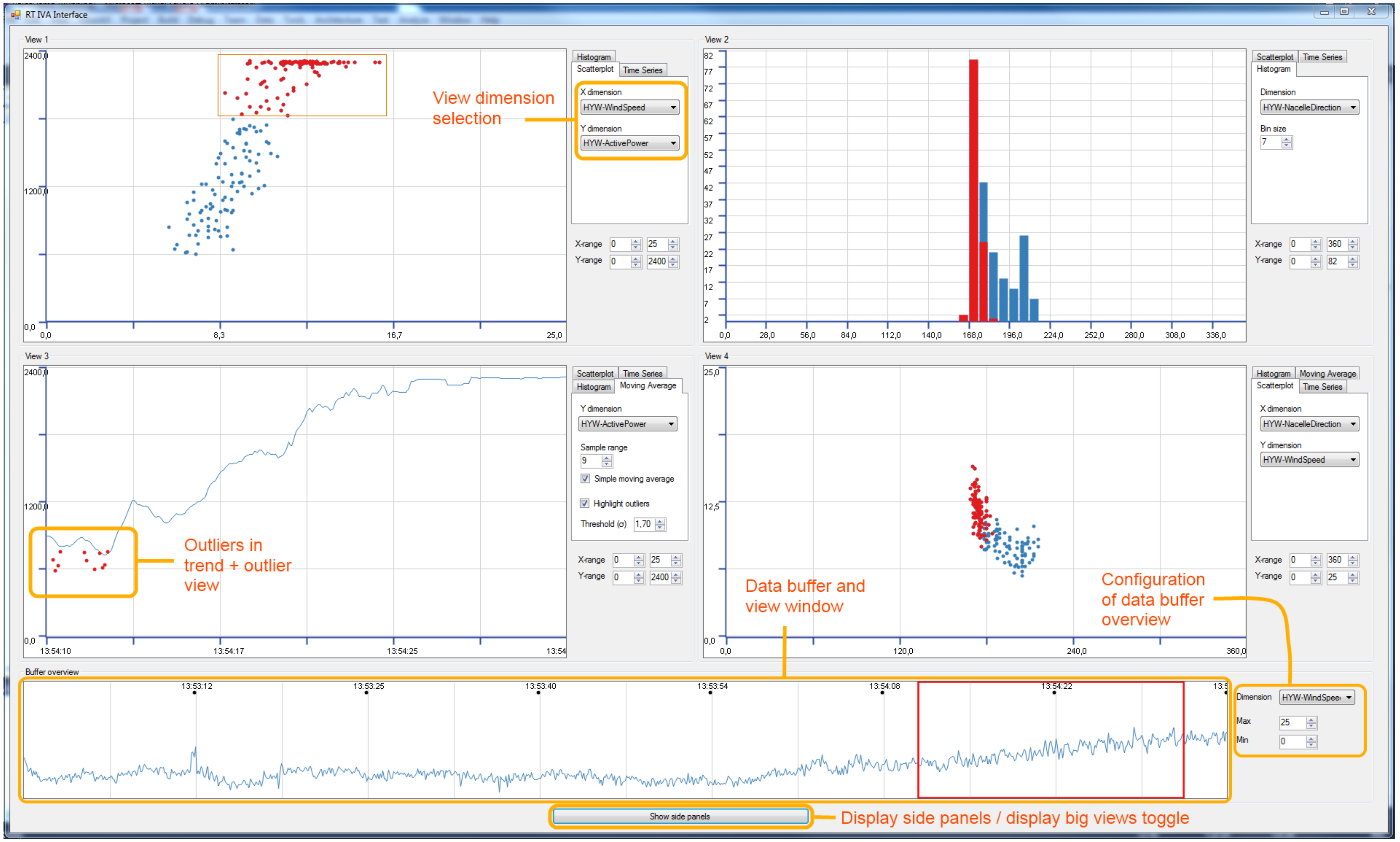
Note: Buffer and archive can only be merged when main buffer is full and archive contains more data points than the main buffer!

Displaying samples 32 - 232
View window size 200
Playback status: Playing, delayed
Buffer size 900
☑ Smooth playback navigation
Data rate 10,0 Hz
☐ Performance mode

Data rate: 0,0 samples/second

View selection
[✓ Head of buffer]
View window
Display last 20 samples

Window range selection
Window range
[✓ Global range]
Fixed
Min 0 Max 100


RT IVA Interface
View 1
View 2
View 3
View 4
Buffer overview
View dimension selection
Outliers in trend + outlier view
Data buffer and view window
Configuration of data buffer overview
Display side panels / display big views toggle
Show side panels

View 1: Histogram | Scatterplot | Time Series
X dimension: HYW-WindSpeed
Y dimension: HYW-ActivePower
X-range: 0 25
Y-range: 0 2400

View 2: Scatterplot | Time Series | Histogram
Dimension: HYW-NacelleDirection
Bin size: 7
X-range: 0 360
Y-range: 0 82

View 3: Scatterplot | Time Series | Histogram | Moving Average
Y dimension: HYW-ActivePower
Sample range: 9
[✓] Simple moving average
[✓] Highlight outliers
Threshold (σ): 1,70
X-range: 0 25
Y-range: 0 2400

View 4: Histogram | Moving Average | Scatterplot | Time Series
X dimension: HYW-NacelleDirection
Y dimension: HYW-WindSpeed
X-range: 0 360
Y-range: 0 25

Buffer overview
Dimension: HYW-WindSpee
Max: 25
Min: 0

Bibliography

[1] Wolfgang Aigner, Silvia Miksch, Wolfgang Müller, Heidrun Schumann, and Christian Tominski. Visualizing time-oriented data – a systematic view. *Computers & Graphics*, 31(3):401–409, 2007.

[2] Wolfgang Aigner, Silvia Miksch, Wolfgang Müller, Heidrun Schumann, and Christian Tominski. Visual methods for analyzing time-oriented data. *IEEE Transactions on Visualization and Computer Graphics*, 14(1):47–60, 2008.

[3] Paolo Angelelli, Kim Nylund, Odd Helge Gilja, and Helwig Hauser. Interactive visual analysis of contrast-enhanced ultrasound data based on small neighborhood statistics. *Computers & Graphics*, 35(2):218–226, 2011.

[4] Statoil ASA. Hywind demo information site. From the Statoil company web site, August 2014. URL http://www.statoil.com/en/TechnologyInnovation/NewEnergy/RenewablePowerProduction/Offshore/Hywind/Pages/HywindPuttingWindPowerToTheTest.aspx?redirectShortUrl=http%3a%2f%2fwww.statoil.com%2fhywind.

[5] Statoil ASA. Sheringham Shoal information site. From the Statoil company web site, August 2014. URL http://www.statoil.com/en/TechnologyInnovation/NewEnergy/RenewablePowerProduction/Offshore/SheringhamShoel/Pages/FactSheet.aspx.

[6] Statoil ASA. Statoil company web site. August 2014. URL http://www.statoil.com/.

[7] Alf Ove Braseth and Trond Are Øritsland. Seeing the big picture: Principles for dynamic process data visualization on large screen displays. In *Proceedings of the*

International Conference on Complexity, Cybernetics and Informing Science and Engineering, pages 16–22, 2013.

[8] Alf Ove Braseth, Øystein Veland, and Robin Welch. Information rich display design. In *Proceedings of the Fourth American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Controls and Human-Machine Interface Technologies*, 2004.

[9] Raphael Bürger, Philipp Muigg, Martin Ilcík, Helmut Doleisch, and Helwig Hauser. Integrating local feature detectors in the interactive visual analysis of flow simulation data. In *Proceedings of the 2007 EuroVis conference*, pages 171–178, 2007.

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

[11] George Chin Jr., Mudita Singhal, Grant Nakamura, Vidhya Gurumoorthi, and Natalie Freeman-Cadoret. Visual analysis of dynamic data streams. *Information Visualization*, 8(3):212–229, 2009.

[12] Wikimedia Commons. Scatterplot of eruption data from the old faithful geyser. From the Wikimedia Commons, August 2014. URL http://upload.wikimedia.org/wikipedia/commons/0/0f/Oldfaithful3.png.

[13] Microsoft Corporation. Overview of the .NET Framework. From the Microsoft Developer Network web site, August 2014. URL http://msdn.microsoft.com/en-us/library/zw4w595w.aspx.

[14] Alan Dix and Geoff Ellis. by chance enhancing interaction with large data sets through statistical sampling. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, pages 167–176. ACM, 2002.

[15] Helmut Doleisch and Helwig Hauser. Smooth brushing for focus+context visualization of simulation data in 3D. *Journal of WSCG*, 10(1–3):147–154, 2002.

[16] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proceedings of the 2003 symposium on data visualisation*, pages 239–248. Eurographics Association, 2003.

[17] Helmut Doleisch, Philipp Muigg, and Helwig Hauser. Interactive visual analysis of hurricane Isabel with SimVis. *IEEE Visualization Contest*, 2004. URL http://vis.computer.org/vis2004contest/vrvis/IsabelWithSimVis.pdf.

[18] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. *ACM Computer Graphics*, 22(4):65–74, 1988.

[19] Geoffrey Ellis and Alan Dix. An explorative analysis of user evaluation studies in information visualisation. In *Proceedings of the 2006 AVI workshop on BEyond time and errors: novel evaLuation methods for Information Visualization*, pages 1–7. ACM, 2006.

[20] Stephen Few. Time on the horizon. *Visual Business Intelligence Newsletter*, pages 1–7, 2008.

[21] Fabian Fischer, Florian Mansmann, and Daniel A. Keim. Real-time visual analytics for event data streams. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, pages 801–806. ACM, 2012.

[22] Tak-chung Fu. A review on time series data mining. *Engineering Applications of Artificial Intelligence*, 24(1):164–181, 2011.

[23] Donna L. Gresh, Bernice E. Rogowitz, Raimond L. Winslow, David F. Scollan, and Christina K. Yung. Weave: A system for visually linking 3-d and statistical visualizations, applied to cardiac simulation and measurement data. In *Proceedings of the conference on Visualization*, pages 489–492. IEEE Computer Society, 2000.

[24] Ming Hao, Daniel A. Keim, Umeshwar Dayal, Daniela Oelke, and Chantal Tremblay. Density displays for data stream monitoring. *Computer Graphics Forum*, 27(3): 895–902, 2008.

[25] David Harrison Jr. and Daniel L. Rubinfeld. Hedonic housing prices and the demand for clean air. *Journal of environmental economics and management*, 5(1):81–102, 1978.

[26] Mark Harrower and Cynthia A. Brewer. Colorbrewer.org: An online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003.

[27] Helwig Hauser. Toward new grounds in visualization. *ACM Computer Graphics*, 39(2):5–8, 2005.

[28] Helwig Hauser. Generalizing focus+context visualization. In *Scientific visualization: The visual extraction of knowledge from data*, pages 305–327. Springer, 2006.

[29] Susan Havre, Elizabeth Hetzler, Paul Whitney, and Lucy Nowell. ThemeRiver: Visualizing thematic changes in large document collections. *IEEE Transactions on Visualization and Computer Graphics*, 8(1):9–20, 2002.

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

[31] Harry Hochheiser and Ben Shneiderman. Interactive exploration of time series data. In *Proceedings of the 4th International Conference on Discovery Science*, pages 441–446. Springer, 2001.

[32] Harry Hochheiser and Ben Shneiderman. Dynamic query tools for time series data sets: Timebox widgets for interactive exploration. *Information Visualization*, 3(1): 1–18, 2004.

[33] Alfred Inselberg and Bernard Dimsdale. Parallel coordinates. In *Human-Machine Interactive Systems*, pages 199–233. Springer, 1991.

[34] Dean F. Jerding and John T. Stasko. The information mural: A technique for displaying and navigating large information spaces. *IEEE Transactions on Visualization and Computer Graphics*, 4(3):257–271, 1998.

[35] Chris Johnson. Top scientific visualization research problems. *IEEE Computer Graphics and Applications*, 24(4):13–17, 2004.

[36] Chris R Johnson and A Sanderson. A next step: Visualizing errors and uncertainty. *IEEE Computer Graphics and Applications*, 23(5):6–10, 2003.

[37] Johannes Kehrer, Florian Ladstädter, Philipp Muigg, Helmut Doleisch, Andrea Steiner, and Helwig Hauser. Hypothesis generation in climate research with interactive visual data exploration. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1579–1586, 2008.

[38] Johannes Kehrer, Peter Filzmoser, and Helwig Hauser. Brushing moments in interactive visual analysis. *Computer Graphics Forum*, 29(3):813–822, 2010.

[39] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, and Hartmut Ziegler. Challenges in visual data analysis. In *Tenth International Conference on Information Visualization*, pages 9–16. IEEE, 2006.

[40] Daniel A. Keim, Florian Mansmann, Jörn Schneidewind, Jim Thomas, and Hartmut Ziegler. Visual analytics: Scope and challenges. In *Visual Data Mining*, pages 76–90. Springer, 2008.

[41] Robert Kincaid. SignalLens: Focus+context applied to electronic time series. *IEEE Transactions on Visualization and Computer Graphics*, 16(6):900–907, 2010.

[42] Zoltán Konyha, Krešimir Matković, Denis Gračanin, Mario Jelović, and Helwig Hauser. Interactive visual analysis of families of function graphs. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1373–1385, 2006.

[43] Zoltán Konyha, Alan Lež, Krešimir Matković, Mario Jelović, and Helwig Hauser. Interactive visual analysis of families of curves using data aggregation and derivation. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies*, pages 24–31. ACM, 2012.

[44] Miloš Krstajić and Daniel A. Keim. Visualization of streaming data: Observing change and context in information visualization techniques. In *Proceedings of the IEEE International Conference on Big Data*, pages 41–47. IEEE, 2013.

[45] Miloš Krstajić, Peter Bak, Daniela Oelke, Daniel A. Keim, Martin Atkinson, and William Ribarsky. Applied visual exploration on real-time news feeds using polarity and geo-spatial analysis. In *Proceedings of the 6th International Conference on Web Information Systems and Technology*, pages 263–268, 2010.

[46] Miloš Krstajić, Florian Mansmann, Andreas Stoffel, Martin Atkinson, and Daniel A. Keim. Processing online news streams for large-scale semantic analysis. In *Proceedings of the 26th International Conference on Data Engineering Workshops*, pages 215–220. IEEE, 2010.

[47] Ove Daae Lampe. *Interactive Visual Analysis of Process Data*. PhD thesis, The University of Bergen, 2011.

[48] Ove Daae Lampe and Helwig Hauser. Interactive visualization of streaming data with kernel density estimation. In *Proceedings of the Pacific Visualization Symposium*, pages 171–178. IEEE, 2011.

[49] Marc Levoy. Display of surfaces from volume data. *IEEE Computer Graphics and Applications*, 8(3):29–37, 1988.

[50] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Pranav Patel. Finding motifs in time series. In *Proceedings of the 2nd Workshop on Temporal Data Mining. At the 8th International Conference on Knowledge Discovery and Data Mining*, pages 53–68, 2002.

[51] Jessica Lin, Eamonn Keogh, Stefano Lonardi, and Bill Chiu. A symbolic representation of time series, with implications for streaming algorithms. In *Proceedings of the 8th Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 2–11. ACM SIGMOD, 2003.

[52] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P. Lankford, and Daonna M. Nystrom. VizTree: a tool for visually mining and monitoring massive time series. In *Proceedings of the 30th International Conference on Very Large Databases*, pages 1269–1273, 2004.

[53] Jessica Lin, Eamonn Keogh, Stefano Lonardi, Jeffrey P. Lankford, and Donna M. Nystrom. Visually mining and monitoring massive time series. In *Proceedings of the 10th International Conference on Knowledge Discovery and Data Mining*, pages 460–469. ACM SIGKDD, 2004.

[54] Vimeo LLC. About Vimeo. From the Vimeo web site, August 2014. URL `https://vimeo.com/about`.

[55] Florian Mansmann, Fabian Fischer, and Daniel A. Keim. Dynamic visual analytics – facing the real-time challenge. In *Expanding the Frontiers of Visual Analytics and Visualization*, pages 69–80. Springer, 2012.

[56] Allen R. Martin and Matthew O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proceedings of the 6th Conference on Visualization*, pages 271–278. IEEE Computer Society, 1995.

[57] Krešimir Matković, Helwig Hauser, Reinhard Sainitzer, and M. Eduard Groller. Process visualization with levels of detail. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 67–70. IEEE, 2002.

[58] Krešimir Matković, Wolfgang Freiler, Denis Gračanin, and Helwig Hauser. ComVis: A coordinated multiple views system for prototyping new visualization technology. In *Proceedings of the 12th International Conference on Information Visualisation*, pages 215–220. IEEE, 2008.

[59] Krešimir Matković, Denis Gračanin, Mario Jelović, and Helwig Hauser. Interactive visual steering – rapid visual prototyping of a common rail injection system. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1699–1706, 2008.

[60] Viktor Mayer-Schönberger and Kenneth Cukier. *Big data: A revolution that will transform how we live, work, and think.* Houghton Mifflin Harcourt, 2013.

[61] Bruce H. McCormick, Thomas A. DeFanti, and Maxine D. Brown. Visualization in scientific computing. *ACM Computer Graphics*, 21, November 1987. Entire issue.

[62] Peter McLachlan, Tamara Munzner, Eleftherios Koutsofios, and Stephen North. LiveRAC: interactive visual exploration of system management time-series data. In *Proceedings of the 26th Annual Conference on Human Factors in Computing Systems*, pages 1483–1492. ACM SIGCHI, 2008.

[63] Steffen Oeltze, Helmut Doleisch, Helwig Hauser, Philipp Muigg, and Bernhard Preim. Interactive visual analysis of perfusion data. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1392–1399, 2007.

[64] Steffen Oeltze, Helmut Doleisch, Helwig Hauser, and Gunther Weber. Interactive visual analysis of scientific data. Tutorial at the IEEE VisWeek, in Seattle, USA, October 2012. URL http://www.ii.uib.no/vis/publications/publication/2012/Hauser12VisTutorial.

[65] Harold Pashler. Dual-task interference in simple tasks: data and theory. *Psychological bulletin*, 116(2):220–244, 1994.

[66] Hannes Reijner. The development of the horizon graph. Presented at the workshop "From Theory to Practice: Design, Vision and Visualization" at the IEEE VisWeek,

in Columbus, USA, 2008. URL http://www.stonesc.com/Vis08_Workshop/DVD/Reijner_submission.pdf.

[67] Jonathan C. Roberts. State of the art: Coordinated & multiple views in exploratory visualization. In *Fifth International Conference on Coordinated and Multiple Views in Exploratory Visualization*, pages 61–71. IEEE, 2007.

[68] George G. Robertson and Jock D. Mackinlay. The document lens. In *Proceedings of the 6th annual ACM symposium on User interface software and technology*, pages 101–108. ACM, 1993.

[69] Takafumi Saito, Hiroko Nakamura Miyamura, Mitsuyoshi Yamamoto, Hiroki Saito, Yuka Hoshiya, and Takumi Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 173–180. IEEE, 2005.

[70] Hans-Jürg Schulz, Thomas Nocke, Magnus Heitzler, and Heidrun Schumann. A design space of visualization tasks. *IEEE Transactions on Visualization and Computer Graphics*, 19(12):2366–2375, 2013.

[71] Ben Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pages 336–343. IEEE, 1996.

[72] Daniel J Simons and Ronald A Rensink. Change blindness: Past, present, and future. *Trends in cognitive sciences*, 9(1):16–20, 2005.

[73] William Szewczyk. Streaming data. *Wiley Interdisciplinary Reviews: Computational Statistics*, 3(1):22–29, 2011.

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

[75] Cagatay Turkay, Julius Parulek, Nathalie Reuter, and Helwig Hauser. Integrating cluster formation and cluster evaluation in interactive visual analysis. In *Proceedings of the 27th Spring Conference on Computer Graphics*, pages 77–86. ACM, 2011.

[76] Jarke J. van Wijk and Edward R. van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 4–9. IEEE, 1999.

[77] Martin Wattenberg. Sketching a graph to query a time-series database. In *Extended Abstracts on Human Factors in Computing Systems*, pages 381–382. ACM, 2001.

[78] Gunther H. Weber and Helwig Hauser. Interactive visual exploration and analysis. Book chapter, to be published. Springer, 2014.

[79] Marc Weber, Marc Alexa, and Wolfgang Müller. Visualizing time-series on spirals. In *Proceedings of the IEEE Symposium on Information Visualization*, pages 7–13. IEEE Computer Society, 2001.

[80] Ian H. Witten and Eibe Frank. *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann, 2005.

[81] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, 23(3):337–343, 1977.

[82] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.