Interactive Visual Analysis of Process Data

Ove Daae Lampe

Dissertation for the degree of Philosophiae Doctor (PhD)

Supervised by Helwig Hauser
Co-supervised by Christopher Giertsen

Institute for Informatics
University of Bergen
Norway

September 2011

To Elena,
Fillip and Patrik

Scientific Environment

This work has been performed at the Visualization group at the Informatics department at the University of Bergen, while beeing an employee at the Computing department at Christian Michelsen Research. I have also been ascosiated with the ICT Research school. This work is part of the research collaboration project eLAD (Petromaks Project 176018/S30, 2007-2011), partnered by International Research Institute of Stavanger, IRIS, Christian Michelsen Research, CMR and the Institute for Energy Technology, IFE. Parts of this research was performed while on a research stay in, and in close collaboration with, the VIDi group at the University of California Davis.

Acknowledgments

First of all I would like to thank my loving wife for the support and understanding I got during my PhD project. I would also like to thank my two kids, Fillip and Patrik, for sacrificing their time for countless evenings (especially close to deadlines) without their dad. I am very grateful to Helwig Hauser for taking me on as a student, and as my supervisor. Hauser has provided me with genuinely helpful guidance through this process, and introduced me into the world of visualization. He has also set the standard for me to strive to follow up on, of which I hope the quality of thesis reflects. I further would like to thank Kåre Villanger and Tor Langeland from CMR in both supporting me in starting with this PhD as well as during this project. I am also very grateful to CMR for providing me with this opportunity, and also for the financial support which made the decision to go "back to school" so much easier for my family to accept. I also acknowledge the financial support for this PhD project from the collaboration project eLAD (Petromaks Project 176018/S30, 2007-2011), partnered by International Research Institute of Stavanger, IRIS, Christian Michelsen Research, CMR and the Institute for Energy Technology, IFE. I would like to thank my parents, Aud Ingrid and Iver Daae Lampe for supporting me, and caring for my family at home during my long hours leading up to deadlines, and when away at conferences. I would also like to thank my brother Claus and his wife Evy-Helen Daae Lampe for being there when I needed to unwind from my PhD work. I especially thank Solvår Hjellestad for being a great inspirational source and who taught me how to love natural sciences through our classes at Rå School. During my visits to Iris in Stavanger, Eric Cayeux was always welcoming, and I would like to thank him for teaching me almost everything I know about drilling. I would also like to thank Gerhard Nygaard at Iris for the support he gave me. Furthermore I would like to thank Mike Herbert at ConocoPhillips for an open and standing welcome to his operations center to observe how they perform their integrated operations on drilling. I am grateful for the opportunity to work at UC Davis, staying in California with my family for three months, given to me by Kwan-Liu Ma of UC Davis. There I also received a great amount of help from Carlos Correa, co-authoring a paper with me, and from Tarik Crnovrsanin in welcoming me to this new city. I would also like to thank Rolf Wilhelm Rasmussen for being my mentor in my early career in the industry, and introducing me to great quality coding, plug-in based architectures and not the least Python. Furthermore I would like to thank: Johannes Kehrer for putting up with me, sharing an office with me all these years, but also for being a good friend and always (OK, almost always)

providing meaningful insights on our research topics. Stig Sandø and Knut Arild Erstad for being good friends and sharing many of my interests. Ivan Viola for proving that academic scientists are cool, when I first entered this world, and for the countless of helpful discussions we have shared since that. Daniel Patel, Endre Lidal, Åsmund Birkeland, Veronica Soltzenova, Paolo Angelelli, Armin Pobitzer and Julius Parulek, Cagatay Turkay, Mattia Natali, Andrea Brambilla and Jean-Paul Balabanian for being good friends, and making my stay at HiB welcoming, interesting and fun. Ole-Morten "Drastiske" Hansen for hours of unwinding nerd activities. Sveinung Sivertsen, Magne and Janne Torsvik, Eivind Samdal, Simon Våge for being good friends. Mark Nijhof for the many nights out at the cinema, and his wife Mona for proofing my English writing. Rune Dalmo for the many hours coding together, producing amazing results and learning OpenGL. Vegar Kleppe who always has interesting feedback on *anything* GPU related, and for critical insight on several new ideas. And, finally, Randall Munroe and Jorge Cham for providing material for procrastination.

Abstract

Data gathered from processes, or process data, contains many different aspects that a visualization system should also convey. Aspects such as, temporal coherence, spatial connectivity, streaming data, and the need for in-situ visualizations, which all come with their independent challenges. Additionally, as sensors get more affordable, and the benefits of measurements get clearer we are faced with a deluge of data, of which sizes are rapidly growing. With all the aspects that should be supported and the vast increase in the amount of data, the traditional techniques of dashboards showing the recent data becomes insufficient for practical use. In this thesis we investigate how to extend the traditional process visualization techniques by bringing the streaming process data into an interactive visual analysis setting. The augmentation of process visualization with interactivity enables the users to go beyond the mere observation, pose questions about observed phenomena and delve into the data to mine for the answers. Furthermore, this thesis investigates how to utilize frequency based, as opposed to item based, techniques to show such large amounts of data. By utilizing Kernel Density Estimates (KDE) we show how the display of streaming data benefit by the non-parametric automatic aggregation to interpret incoming data put in context to historic data.

Contents

Related Publications

This thesis is based on the following publications:

A. O. Daae Lampe and H. Hauser. **Interactive Visualization of Streaming Data with Kernel Density Estimation**. In *Proceedings of the IEEE Pacific Visualization Symposium 2011*, pages 171–178, Hong Kong, March 1–4, 2011.

B. O. Daae Lampe and H. Hauser. **Curve Density Estimates**. In *Proceedings of Eurographics/IEEE-VGTC Symp. on Visualization (EuroVis 2011)*, 30(3), pages 633–642, Bergen, Norway, June 1–3, 2011.

C. O. Daae Lampe, C. Correa, K. L. Ma and H. Hauser. **Curve-Centric Volume Reformation for Comparative Visualization**. In *IEEE Transactions on Visualization and Computer Graphics (IEEE Vis. 2009)*, 15(6), pages 1235–1242, 2009.

D. O. Daae Lampe, J. Kehrer, and H. Hauser. **Visual analysis of multivariate movement data using interactive difference views**. In *Proc. Vision, Modeling, and Visualization (VMV 2010)*, pages 315–322, 2010.

E. O. Daae Lampe and H. Hauser. **Interactive Model Prototyping in Visualization Space**. In submission to *SIGRAD 2011 in Stockholm, Sweden.*

The listed papers were all written during the Ph.D. project of the thesis author. The thesis author is also the main author of all the publications. Furthermore, all papers where co-authored by the supervisor of this thesis, Helwig Hauser. Hauser provided, through guidance, inspiration for most of the novel contributions presented here. Paper C where co-authored by, then post doc. at the VIDI group at UC Davis, Carlos Correa, and Kwan-Liu Ma, Professor of the VIDI group at UC Davis. This paper was written during a research stay at the VIDI group, and the ideas and contributions where formed through a tight collaboration between the authors. Carlos Correa contributed in the implementation and research phase by contributing data and with insights into volumetric deformations. In the writing phase of the paper, Correa wrote the largest part of the related work section, where the rest of the paper was written to the greater extent by the main author. Paper D was also co-authored by Johannes Kehrer who helped with several aspects of the interactive difference views, and with the write up.

Part I

Overview

Chapter 1

Introduction

The interactive visual analysis of process data, or interactive process visualization, is the combination of traditional process visualization and techniques from interactive visual analysis. This thesis describes visual representations that support both streaming process data, and visual interaction techniques. To a greater extent than ever before we gather data using sensors monitoring complex systems, ranging from run sensors that monitor heart-rate, speed and position to monitor your training progress, to sensors placed on every cab in large cities, and to traditional process sensors monitoring the flow of different materials in a large chemical production plant. With an increased connectivity and a steep drop in price, the placement of sensors is becoming more and more ubiquitous, and their data can to a greater extent be gathered and utilized. What all these sensors have in common is their ability to monitor and report a value measured from some physical process. All these sensors also have a physical location, which might change over time, and they report their values given a timestamp which combined fixes their relevance in time and space. In some instances the sensors produce spatially connected and continuous values, and in some they do not, such as across boundaries. However, for almost all sensors monitoring physical processes they produce continuously connected values over time. Addressing the increased amount of sensors, the individual sensors abilities to produce more and more data and the widespread connectivity enabling the relay of this data to central servers; we see the need for new techniques to monitor their data. Traditional process visualization often consist of dashboards made to convey the status quo of the process, predominantly also made to show the physical location of the different sensors. From the cognitive and psychological viewpoint these dashboards are made to convey information as good as possible to the operators, while carefully avoiding overloading them. The amount of information an operator can handle is tightly coupled to the *pace* this operator is working in. Figure 1.1 illustrates how pace and the amount of information an operator can consume is connected. Where an analyst can interact with the data, making an educated guess, a firefighter would need a tight selection of the most important data, e.g., temperature or target location, and then without any interaction act instinctively. Visual analytics frameworks, on the other hand, are made to fully exploit interaction and are usually made with as many options as possible to fully analyse any number of situations. Figure 1.1 illustrates visual analytics

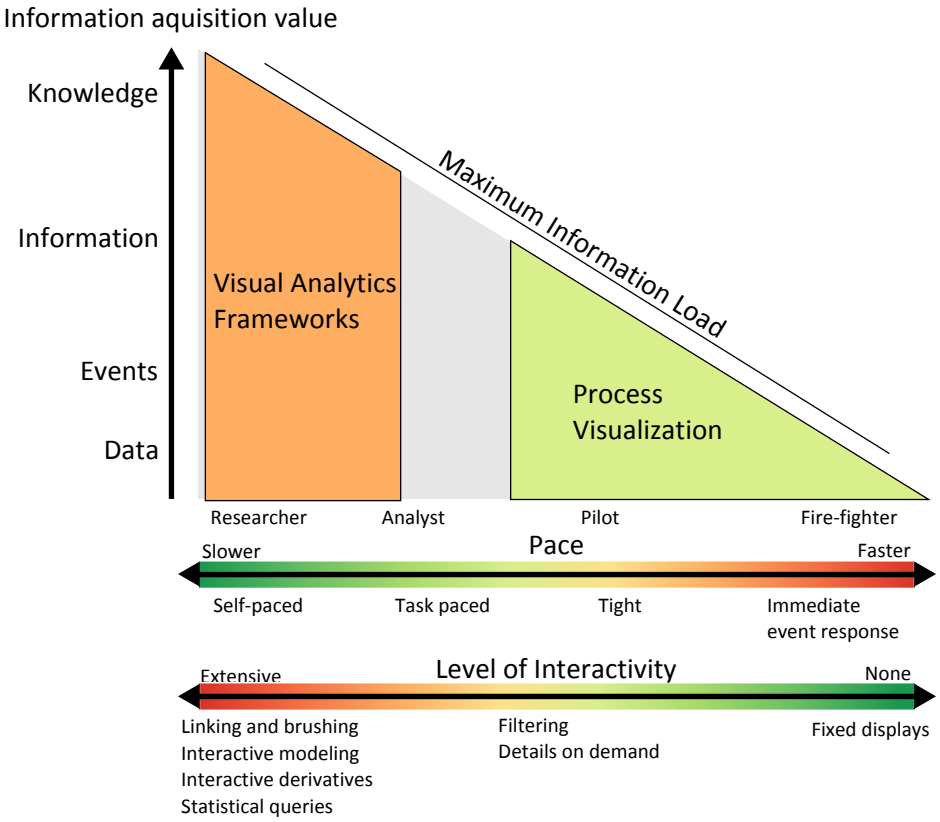


Figure 1.1: Figure showing the pace (time to solve a given task) and interactivity needed to increase information value.

frameworks in the other end of the temporal scale, indicating that they do take more focus and time to work with. With interactive process visualization, as coined from this thesis title, we intend to extend the range of process visualization towards that of visual analytics, thus, enabling the changing paces, or roles a process engineer must take.

1 Problem Statement

The motivation for this thesis was found through the eLAD project [91] in close collaboration with the involved industry partners. The eLAD project is supported by the research council of Norway and made up by Statoil and ConocoPhillips as industry partners, and Iris, IFE and CMR as research partners. The main

challenges which this thesis should address were related to the introduction of the wired pipe [16] technology into the drilling process and to facilitate integrated operations. The wired pipe technology widens the bandwidth of communication with sensors in the well (while drilling) from approximately seven bits per second, using the current technology, to several megabit/s. This increased bandwidth enables a large amount of new sensors to be developed and used, flooding the current workflow with data. The second challenge lies in supporting integrated operations, meaning, supporting the integration of the teams working offshore at the oilrig with the land based operation center. From this we extracted two major visualization challenges. First, to support the different paces for the operators (as illustrated in figure 1.1), which differ greatly with respect to the driller at the rig, and as compared to those who operate the onshore operations center. The visualization need visual representations that work in both a static and in an interactive setting. Second, accounting for either the massive increase in data from current sensors, or a large increase in different sensors, the visualization techniques should work on streaming data in an in situ setting.

2 Contributions

In this thesis, in particular in chapter 3 of part I and in the papers of part II, we describe the following novel contributions:

1. A line kernel to represent continuous data in density/frequency based visualizations.
2. An interaction technique for KDE based visualizations, connecting the bandwidth to the zoom level.
3. Differential analysis enabled by interactively defined difference views.
4. A new and easily reproducible visualization technique that can display single or multiple curves irrespective of frequency and zoom.
5. Curve-centric volume reformation, a technique to align 3D volumetric data to a 1D parametric curve, which also includes the definition of a modified Frenet frame and a realtime inside-out raycasting.
6. A technique for interactive model prototyping, which includes a novel detrending of the value space to provide feedback on the quality of fit.

3 Thesis Structure

This thesis is divided into two main parts. This first part summarizes and abstracts the findings that were worked out through this project and documented in several individual publications. In the second part these individual publications follow. These papers are provided ad verbatim, in their published form, with the

exception of their style, which is conformed to fit this thesis, and their bibliographies which were collated. Paper E will, naturally, undergo some changes, since it is still in submission.

In this part we provide a chapter covering work related to our contributions. This chapter is a supplement to the more detailed and specific related work sections found in the individual papers in part II. Following, in chapter 3, a detailed look into the novel contributions made in this thesis is described. Then, a demonstration of the applicability of the contributions, with particular focus on the related industry, is made in chapter four, before a conclusion is provided in chapter five.

Chapter 2

Related Work on Interactive Visual Analysis of Process Data

This chapter provides a brief overview of work related to interactive visual analysis of process data. To cover this combined terminology, we first cover process data, and the visualization thereof, before focussing on the streaming aspect. We then go into interactive visual analysis and look at several concepts there. Two other important aspects, covered in this thesis, are comparative visualization and kernel density estimation, which are both also covered here.

1 Process Visualization

Usually, the main purpose of process visualization is to enable an operator to monitor the status of a process, and to support decision making. A traditional operator is often given the role of a *process pilot*. Braseth et al. [18] recognized that this role changes during the course of operation. When, e.g., unexpected problems arise, their role might more closely resemble that of a fire-fighter. When the problems are resolved, the operator might even adopt a researcher role to find out what went wrong in the first place. The decisions handled by the operator are ranging from the small subconscious ones to more deliberate ones. How these decisions are made, defined by Rasmussen [96] as the SRK framework, can be separated in three levels of control, namely: skill based (S), rule based (R) or knowledge based (K).

- The skill based behavior is the low level response to observations, processed continuously from a stream of data. Examples of skill based behavior are maintaining the position of a car in a lane, or maintaining the pressure level and rotation speed in a drilling operation. This behavior is mostly subconscious, and can thus be maintained with minimal attention. We have a high parallel capacity for such tasks.
- Rule based behavior is defined as the response to "familiar" situations, where an event should be interpreted, by pattern recognition, and the normal response should be initiated. Two examples of such behavior is stopping the car at a red light, or shutting down a pump when the receiving tank is full. The parallel capacity for such behavior is moderate.

- Knowledge based behavior, or problem solving, is the complex process of gathering information, integrated from multiple sources, to formulate a plan and to execute the proper response. This response should be based on as much relevant information as possible, to get the full picture, such that the operator can relate to previous experiences or knowledge about the processes. This behavior is mentally demanding and usually requires the operator's full attention, such that parallel tasks are detrimental to the outcome.

Braseth et al. [18] recognized the temporal aspect in these different roles, and the maximum information load possible, which is showed in Fig. 1.1.

The cognitive and psycological sciences defines several different methodologies of *interfaces design*. One methodology is user-centered design, which focuses primarily on the tasks to be performed and the operator that should perform them. A differing methodology is the ecological interface design (ECI) which was introduced by Vicente and Rasmussen [127]. ECI sets focus at the work-domain and the environment. ECI builds upon two main concepts, the abstraction hierarchy [97], which defines how to model the work environment, and the SRK framework which model the operator's different roles. Since its introduction ECI has seen several implementations ranging from less complex processes, to large and complex process monitoring [134], such as nuclear plant management [17].

2 Streaming Data

With an increasing ability to collect data in almost all fields, we find ourselves with data growing faster than our ability to process and analyze it. The prevalent technique, both in statistics and visualization, of storing and then analyzing in-place, is often unusable in the context of streaming data. Szewczyk defined streaming data [114] as a continuous sequence of ordered observations of indeterminate length. This definition, while simple, carries some important implications. Since the data arrives continuously and there is no known end to it, one cannot read all of it, and then process it. Rather, one must process the current data, and then eventually, before one runs out of memory, discard old data to fit the new. If either the old data is not removed, or if the process stalls, e.g., due to a too high CPU load during analysis, newly arriving data will simply be lost.

Szewczyk recognized three factors that define streaming data, one, the continuously arriving data, two, the unknown sample size, and three the permanent loss of data if not explicitly kept and stored. Aside from the obvious limitation of storage space, the defining factor that determines if streaming data can be analyzed and visualized is the time spent to process new samples. Unless every new sample can be analyzed in real-time, i.e., faster than the samples are arriving, new samples will either have to be discarded, or left out from the analysis. Processing time, in other words, is bounded by acquisition time. Process visuali-

zation, and data from processes, thus cannot automatically qualify as streaming data, since often the size can be manageable in terms of storage. This data does, however, share the other two aspects of streaming data, namely, continuously arriving data and an unknown sample size. However, for many practical applications for real-time process visualization the limitation on storage space still applies, since process visualization is often either done, offline, on the historical stored data, where time is not an issue, or directly on the stream. Algorithms that are applicable to streaming data need to first be applicable to data with an unknown size, and second fit within the limitation on processing time. An example of an algorithm that is not applicable to streaming data is the following implementation of the sample mean, $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, which needs the full dataset $[x_i]_{i=1,\ldots,n}$. This algorithm can be rewritten as a recursive algorithm, $\bar{x}_n = \bar{x}_{n-1} + \frac{1}{n}(x_n - \bar{x}_{n-1})$, and can thus be applied to a stream, by only storing $\bar{x}_n$.

Wong et al. [139] showed how the Haar wavelets can be applied to streams, to compress large data amounts, without loosing important features. Wong et al. [139] also investigated how to perform PCA and MDS on streams, where the eigenvectors are only calculated for parts of the data, while the rest are projected using the same eigenvectors as the first part. Zhou et al. [145] presented M-Kernel Merging (MKM), to evaluate the kernel density estimation on streaming data. MKM works by applying binning to the incoming samples, and these bins are represented as a single kernel weighted by the bin count. When the total bin count exeeded a given threshold, two bins would be merged and a new bandwidth and weight would be calculated for the resulting bin. Considering that the kernel density estimation can already be defined for a fixed evaluation grid in a recursive term, and thus being applicable for streaming data, MKM enables both dynamic grids, and a kernel size/bandwidth that can change along the stream.

Applications, algorithms and visualizations that operate on streaming data can, in our opinion, be separated in two distinct forms: one which operates solely on the current window[1], and the other which operates on persistent results. Where windowed algorithms, in this sense, only contains information extracted from the current window, a persistent algorithm will contain information which evolved over the entire stream. As an example of a window based algorithm, consider a window of the ten latest samples, and their average, calculated in full whenever one sample enters (and one leaves) the current window. As an example for a persistent algorithm, we consider the recursive average, where, whenever a new sample arrives, the existing average is updated (using the recursive algorithm above), and will thus hold information about the entire stream.

Other methods to increase the persistence of the stream includes decimation, aggregation, modeling and abstraction. A decimation technique, could by uti-

[1] A windowed function, or an apodization function, is a mathematical function that remain constant in a given interval, and zero outside. In streaming data algorithms, this usually mean the recent subset of samples, that we can effectively handle.

lizing statistical sampling reduce the amount of samples that are kept, or use the Haar wavelets to compress the stream, as proposed by Wong et al. [139]. Aggregation techniques summarize several samples to reduce both the size and complexity. BinX [14] employed temporal binning at different levels of aggregation while calculating the mean and higher moments. Aggregation could also be applied to other measures than the temporal axis, e.g., count the hits to a web server per city. Modeling techniques can by understanding the underlying rules/physics that govern the process which the measured data is coming from, extract information on a higher level, e.g., discard all data from a "normal" operation, and show model parameters, instead of raw data, for the rest (as shown in paper E). Abstraction includes the creation of higher concepts, often of higher value than individual samples, from the data stream. Detecting and storing an event with extracted details, e.g., an ongoing denial of service attack, is an example of abstraction. Abstractions are often more closely related to the mental model we use, i.e., looking at millions of SYN requests (spawning half-open TCP connections), as one single DOS instead of individual requests.

Hao et al. [52] utilized a pixel based displays to show a time window into streaming data. Since streaming data can come in at high speeds, Hao et al. also investigated different layouts to minimize the overall movement of the display, when introducing new data.

3 Interactive Visual Analysis

Thomas and Cook [117] defined the field of visual analytics as following:

> Visual analytics is the science of analytical reasoning facilitated by interactive visual interfaces.

And as further defined by Keim et al. [72], visual analytics is a highly interdisciplinary field ranging from information analytics and statistical analytics to cognitive and perceptual sciences. Direct interaction with both the data and visual representations are key elements in In interactive visual analysis (IVA). Shneiderman coined the visual information seeking mantra [106] for what he recognized as a repeating pattern for the common interaction while seeking information in data. Keim et al. [71] found that when analysing large amounts of data, an overview might not be an option. Keim et al. further expanded on Shneidermans mantra, by including automated analysis, to:

> Analyze First, Show the Important, Zoom, Filter and Analyze Further, Details-on-Demand

In the following sections we will describe work related to two important aspects of IVA: interactive visual analysis and exploration, interaction techniques and visual representations.

3.1 Interactive Visual Exploration and Analysis

In 1977 Tukey [123] presented in the seminal work on *exploratory data analysis*. Up to that date, much of the statistical visualization consisted of static figures of results. Tukey suggested to connect the visualizations, using interaction, directly to the data. In visual exploration, and particularly when dealing with large multidimensional data, one are not looking for the answer to one particular question, but rather picking up evidence during the interactive exploration. This evidence, important pieces of information or findings, was termed by Yang et al. [142] as *nuggets* of information. Liu and Stasko [80] investigated how the internal mental model, or nuggets, relate to external visualizations. Yang et al. [142] showed that providing techniques to externalize these nuggets back into the visualization enables both reasoning and collaboration. This externalization was provided through their *nugget management system*, where the nuggets could be added as evidence for, or against, a larger proof or hypothesis. Shrinivasan and van Wijk presented the *Knowledge View* [107] that enabled externalization through a mind map, where the nodes where linked to the interaction steps needed to reproduce the externalized findings.

3.2 Interaction and Multiple Coordinated Views

Interaction is one of the primary distinctions between regular analysis and IVA. The role of interaction, as a mean of connecting the user to the data, was classified by Yi et al. [143] based on the users *intent*. They identified the following seven intentions that the user had with respect to the data and its visual representations:

- *select*, to mark something as important,
- *explore*, show me something different,
- *reconfigure*, show me a different arrangement,
- *encode*, show me a different representation,
- *abstract/elaborate*, show me more or less detail,
- *filter*, show me something conditionally,
- *connect*, show me related items

A central method for the interactive exploration/analysis of multivariate data, is that of linking & brushing, compare to Eick and Wills [33] or Tweedie et al. [125]. Brushing is the defintion of a selcection, or marking a subset of data elements as interesting. Linking is the highlighting of the corresponding subset in other views, of the same data, to highlight their relations. This highlighting is often performed utilizing techniques described as *focus+context*, detailed by Hauser [55], which presents how to emphasize a subset, the *focus*, while maintaining the overview, the *context*.

Coordinated multiple views is a methodology that use multiple views on the same data, and has these views connected. This connection is often realized using linking and brushing, or other focus+context methods. There are several applications that support this visual querying, e.g., the XmdvTool [132], the SimVis framework [31], Spotfire® [1], Tableau® [115] or ComVis [83].

4 Comparative Visualization

Pagendarm and Post [92] separated comparative visualization where images are placed side by side, leaving the interpretation of differences to our cognitive system, from that of direct comparative visualization. In direct comparative visualization the resulting image shows the evaluated difference in one target image. Pagendarm and Post [92] furthermore identified two primary approaches of performing direct comparative visualization, the first as image level comparison, and the second as data level comparison. In image level comparison two images, resulting from their own visualization pipelines, are overlaid using either a blending or difference operator. In data level comparison, data from two different sources are converted to a common representation and then both are fed into the same visualization pipeline. Later Verma and Pang [126] added feature level comparsion to this list of primary approaches. In feature level comparison, different methods of extracting similar features from the same dataset are overlaid into the same image, highlighting the differences in a direct manner.

In paper C in part II of this thesis, different data-sources are visualized using different visualization pipelines, and as such cannot be called a direct comparative visualization. However, in combination with a proposed helper line and a shared parallel axis, a direct comparison can be made in the one shared dimension. In paper D a direct comparative visualization is used to show the differences between two density fields representing two different categories. This difference operator is implemented on the evaluated kernel density estimation, but before the visual representation, and is thus partly between the data level and the image level comparative visualization approaches. To visualize the result from a difference operator, which also can yield negative values, a diverging colormap, compare to [19], is applied.

5 Kernel Density Estimates in Visualization

Kernel density estimation, KDE, has a long history dating back to its introduction first by Rosenblatt in 1956 [101] and later, in 1962, independently also by Parzen [93], after which it also recieved its name as the Parzen window, or also as the Parzen-Rosenblatt window.

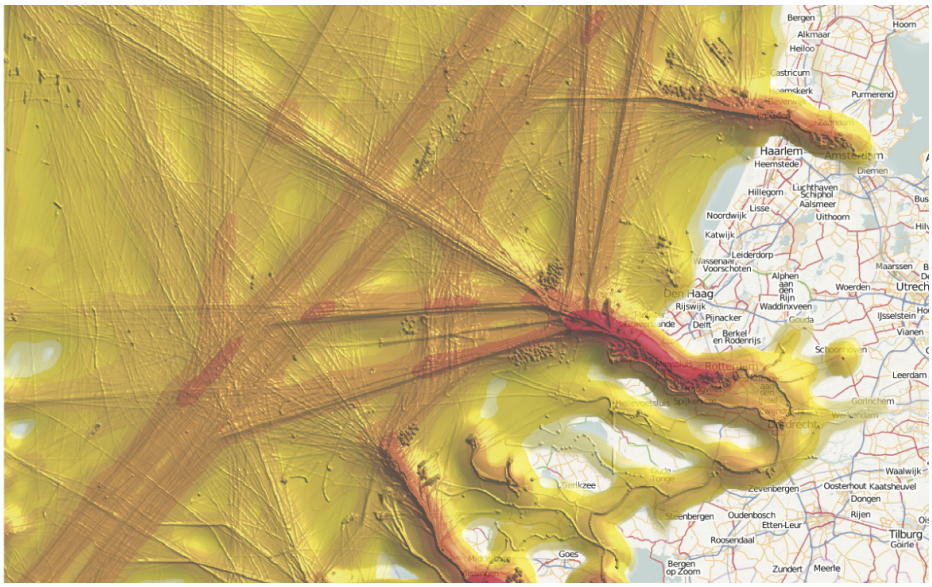


Figure 2.1: Visualization of vessel movements, courtecy of Willems et al., showing how to combine two different bandwidths (kernel sizes), to show overall density in addition a more detailed view.

There is a tight connection between density estimation and visualization, since often the density estimation is created to inspect a distribution visually. Scott wrote a good overview of multivariate density estimation in visualization [104]. Different density maps, similar to those of KDE, have been used in cartography for several applications. Fisher introduced a technique called hotmap [39] to visualize the frequency of views of different map tiles. In a similar application, Whittaker and Scott presented the use of the average shifted histogram [135], as an effective approximation for KDE. Also for a geographic use, Willems et al. introduced KDE to visualize vessel movements [136]. In this work they combined the usage of two different kernel sizes, to first show the overal density of vessel movements, and secondly to show detailed movements, as also shown in figure 2.1. Later, Scheepens et al. [103] refined this work to include a GPU pipeline for rendering, and tools to enable multivariate analaysis.

Minnotte and Scott presented the mode tree [86] which is a visualization of the continiously changing modes when the bandwidth is increased. The mode tree was further refined by Minnotte et al. [85] to move in the direction of continous representation of modes, constructed from multple distributions. Florek and Hauser [42] presented an improved technique on how to visually analyze the

bivariate mode tree. Florek and Hauser [41] also investigated how to improve the quantitative visual properties of bivariate KDE through indicators and iso-lines.

As very similar to KDE, we can also consider radial basis functions (RBF). Jang et al. investigated the representation of volumetric datasets by weighted radial basis functions (RBF) [61, 21], and introduced an effective algorithm on how to render these. Crawfis and Max [26] used KDE to reconstruct uniformly sampled 3D volumes.

Chapter 3

Interactive Visual Analysis of Process Data

With the aim to support process engineers to switch from the role of process pilots to the role of an analyst, or even a researcher, we have defined several methods that support these tasks. In the remainder of this thesis, these methods are presented as the following novel contributions:

1. A line kernel to represent continuous data in density/frequency based visualizations (related publication: Paper A in Part II).
2. An interaction technique for KDE based visualizations, connecting the bandwidth to the zoom level (Paper A).
3. Differential analysis, enabled by interactively defined difference views (Paper D).
4. A new and easily reproducible visualization technique that can display single or multiple curves irrespective of frequency and zoom (Paper B).
5. Curve-centric volume reformation, a technique to align 3D volumetric data to a 1D parametric curve, which also includes the definition of a modified Frenet frame and a realtime inside-out raycasting (Paper C).
6. A technique on interactive model prototyping, which includes a novel detrending of the value space to provide feedback on quality of fit (Paper E).

The following sections in this chapter are structured as following: first we cover the progress we have made in improving kernel density estimates (KDE), and then we cover the novel contributions on interactive techniques in KDE, before going into the concept of curve-centric volume reformation.

1 Kernel Density Estimates for Continuous Data

In the papers D, B, A and E (in part II of this thesis), a common theme has been the usage of kernel density estimation (KDE) to investigate temporal data. A big difference between the existing KDE and our focus towards temporal data, is the connectivity between the samples. E.g., samples from a satisfaction survey yield discrete samples (individual customer opinions), but a sampled time-series of a measurement from a physical process should be interpreted as snapshots of a continuous change. In the following we first provide a brief and general

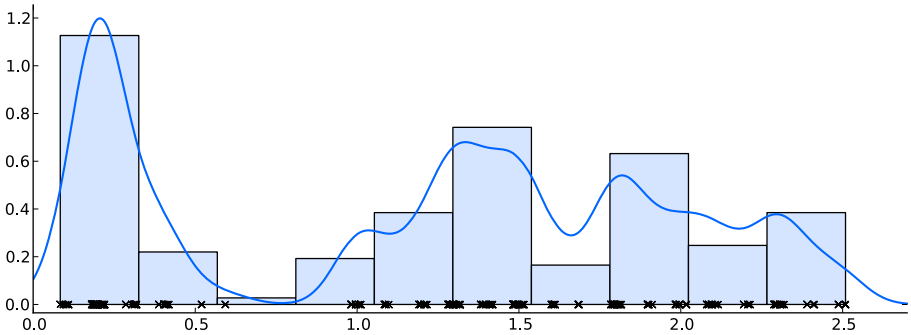


Figure 3.1: Data samples from the Fisher Iris dataset [40], shown as a rug plot (with a small random displacement to also show overlapping values), as a normalized histogram and as a one-dimensional kernel density estimate (the blue line).

introduction to visualizations based KDE, followed by a description of our novel contributions.

1.1 Kernel Density Estimation

A short description of a kernel density estimate (KDE) is to consider it as a continuous histogram. Instead of defining a discrete number of bins and counting how many samples that belong to each of them, a KDE is defined as the sum of a series of kernel instances. Each of these represents one of the samples in the original data set. Fig. 3.1 shows an example KDE, compared to the corresponding histogram. While these two visualizations correspond well here, we show how histograms can suffer from aliasing effects in paper A.

KDE was introduced by Rosenblatt and Parzen over 50 years ago [101, 93]. Given a set of n (1D) data samples x_i the kernel density estimator $\widehat{f}_h(x)$ is computed as

$$\widehat{f}_h(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i), \tag{3.1}$$

based on a kernel function K and a bandwidth parameter h. While KDE is defined for an arbitrary kernel function, K, we have primarily focused on the normal distribution. From the KDE in its one-dimensional form (Eq. 3.1), we extend to the N-dimensional form (as shown by Scott [104]) by

$$\widehat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) \tag{3.2}$$

with $\mathbf{H}$ being a symmetric and positive definite bandwidth matrix and $K_{\mathbf{H}}$ being defined as

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}} \mathcal{K}(\mathbf{H}^{-\frac{1}{2}}\mathbf{x}).$$

$\mathcal{K}$ is a multi-variate kernel function that integrates to 1. This equation will for a set of samples, $\mathbf{x}_0, ..., \mathbf{x}_n$, create a continuous estimate of density, i.e., a probability density estimate with the unbounded integral of one.

Since the individual kernels integrate up to one, the normalization term of Eq. 3.2 is $1/n$. If we remove this term, the unbounded integral of Eq. 3.2 is n. In paper A we show different cases when this is desirable. E.g., if the individual samples represent a dollar spent at a given location, then the resulting KDE will show the distribution of dollars spent over the total area. In the same example, a bounded integral will result in the given amount spent within that bounded area[1]. To facilitate these desirable properties of the KDE we iterated on equation 3.2 (in paper A). We removed the normalization, and introducing a scaling factor per kernel, c_i. To revisit our previous example, the scaling factor allows for a single sample/kernel to represent multiple dollars, instead of a single dollar. The iterated equation for this 2D KDE is

$$\widehat{g}_{\mathbf{H}}(\mathbf{x}) = \sum_{i=1}^{n} c_i K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i). \tag{3.3}$$

This new definition enables the aggregation of other attributes than just sample density, e.g., time in seconds, as shown in later examples, or dollar per square mile as illustrated in Fig. 3.2. This figure, 3.2, illustrates individual contributions as samples, each scaled by the contributed amount, c_i. As opposed to histograms and normalized density estimations the scale can also be negative. In the case of Figure 3.2 , much to our surprise this contribution dataset also contained negative amounts. These negative amounts represented the contributions that were rejected and sent back, and there are, in several areas, more geospatially located rejections than acceptions (shown as blue).

1.2 The Line Kernel

For most sensors utilized to measure physical processes, a value is read out at discrete intervals and streamed to its host. Most physical processes, however, are continuous in nature, which means, e.g., if a temperature is measured at five degrees one second and at ten degrees the next second, it is more correct to represent this change with a smooth continuous change from five to ten, than a discrete jump. To represent the density of temporally connected samples given

[1] In actuality the bounded integral, will not yield exactly the same result as the sum of the samples. KDE distributes each sample over a small area, and thus, samples outside the bounded area can contribute in, and vice versa.

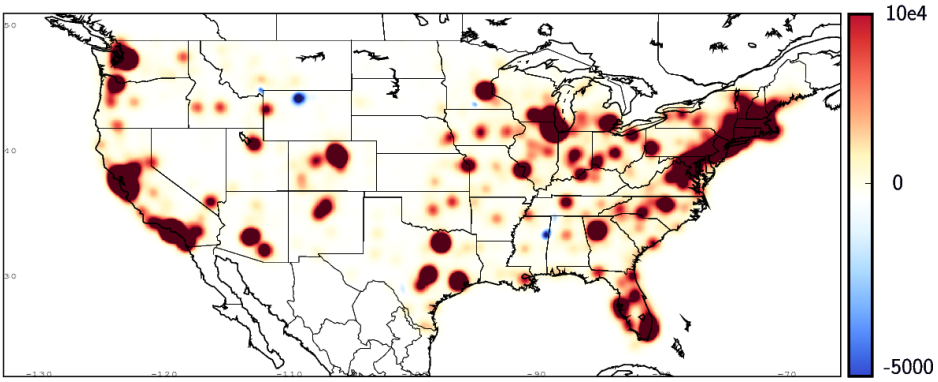


Figure 3.2: Visualizing over 165 000 monetary contributions to the Obama 2008 campaign. Interesting areas with negative aggregates, i.e., locations where the amount of contributions are less than zero dollars per square km (shown as blue).

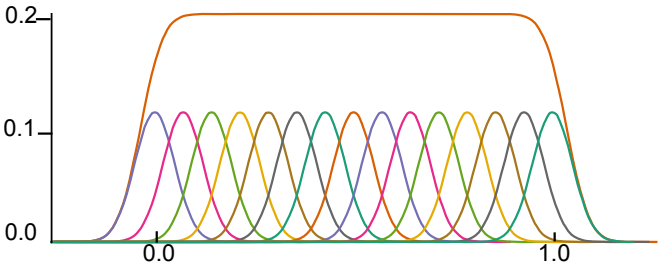


Figure 3.3: The line kernel in 1D, shown as upper orange curve, is given as the integral of individual kernels along the line between its start (here 0) and its end (here 1), is defind by Eq. 3.4.

over a specific domain one can either resort to linear interpolation and subsampling or, as we argue for in this section, utilize a specialized line kernel. In papers A and B of part II, we show several cases of the beneficial properties of this line kernel.

The line kernel accomplishes two main advantages over drawing a regular line. The first advantage is its normalization; where a regular line will draw more pixels when the line gets longer, and thus add more to an integral or sum, the line kernel is always normalized. The second advantage is the singular degeneration a line will have if the two points fall on the same pixel. A regular line drawing will draw zero pixels in this case. The line kernel, however, converges towards a regular 2D normal distribution when the two samples converge.

In paper A, we describe this as a line kernel L_k defined by two consecutive data samples, and their positions $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$:

$$L_k(\mathbf{x}) = \int_0^1 c_i K_{\mathbf{H}}\big(\mathbf{x} - ((1-\phi)\mathbf{p}_i + \phi\mathbf{p}_{i+1})\big)\,\mathrm{d}\phi, \tag{3.4}$$

with $K_{\mathbf{H}}$ being the 2D normal distribution kernel (or any other proper 2D kernel). This kernel, L_k, as shown as a 1D function in Figure 3.3, is the integral of a series of small kernels moving from $\mathbf{p}_i$ to $\mathbf{p}_{i+1}$. To enable a proper reconstruction from uneven sampling in time, we insert the elapsed time between the two samples in the scaling factor c_i. The realization of this integral, in paper A, was achieved by using preintegrated look-up tables for rendering, but in paper B we defined a direct and analytical definition. To find this analytical definition, we first define a 1D version of equation 3.4. This 1D version is defined by reducing the dimensionality of the kernel to 1D, also shown in figure 3.3. The analytical definition is then found by considering a single point $x < p_0$ with $p_1 \approx \infty$ of this 1D version of Equation 3.4. The evaluated value for this x is the integral of an infinite series of normal distributions with their mean increasing from p_0. By turning this definition around, we observe that this is equal to the bounded integral of a single normal distribution with mean p_0 from $-\infty$ to x. The integral of the normal distribution is a cumulative distribution function (cdf). This distribution function is defined by

$$\mathrm{cdf}(\mathrm{x}, \mu, \sigma) = \frac{1}{2}\left(1 + \mathrm{erf}\big(\frac{x-\mu}{\sqrt{2\sigma^2}}\big)\right), \tag{3.5}$$

with erf the error function, found when integrating the normal distribution,

$$\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}} \int_0^z e^{-t^2}.$$

When considering $p_1 < \infty$ we have to remove the contributions of all kernels larger than p_1, which leaves:

$$L_{k1D}(x) = \frac{1}{|p_1 - p_0|}\big(\mathrm{cdf}(\mathrm{x}, \mathrm{p}_0, \sigma) - \mathrm{cdf}(\mathrm{x}, \mathrm{p}_1, \sigma)\big). \tag{3.6}$$

This 1D line kernel is then expanded back to 2D by first creating a new parameterization with u extending along the line segment and v orthogonal to it[2], and then by applying the following product with the normal distribution kernel:

$$L_k(\mathbf{x}) = c_i L_{k1D}(u) \cdot N(v) \tag{3.7}$$

Figure 3.4 shows a single line kernel of the line segment between points $[0, 0]$ and $[1, 1]$, and the parameterization directions for u and v as utilized in Eq. 3.7.

[2]more details on this parameterization in paper A in part II

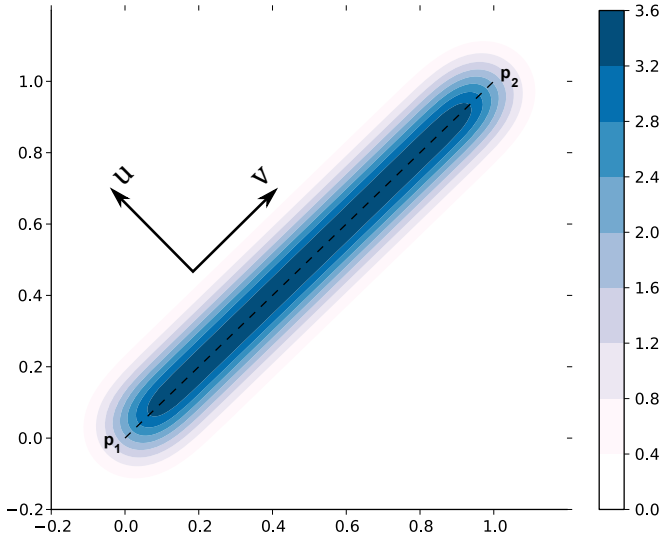


Figure 3.4: A 2D line kernel, as defined in Eq. 3.7 with $\mathbf{p}_0 = [0, 0]$, $\mathbf{p}_1 = [1, 1]$ and $c_i = 1$. The integral of this line is equal to c_i, i.e., 1.

1.3 Curve Density Estimates

If we graph a sine curve from zero to, e.g., a thousand π, depending on our screen resolution, the result will resemble an opaque square spanning the full range, as shown on the top of figure 3.5. A common solution for coping with such situations of massive overdraw is to apply transparency. The second graph in figure 3.5b shows the same graph where semi-transparency is applied. In this graph, figure 3.5b, it seems like there is a higher density at $y = 0$. If we, however, take regular samples and plot them using a scatterplot, the result looks quite different, as shown in the third figure from the top (c). In fact, if we take regular samples at x from this sine curve and insert their y value in a histogram, we see the direct resemblance towards this scatterplot, as illustrated in Fig. 3.6. To reintroduce connectivity, but keep this visualization of density we introduce curve density estimates.

The curve density estimate (CDE) is the continuous probability density estimate of sampled points along a curve. As shown in figure 3.5e the CDE displays the continuous density over y while also representing a continous curve, as opposed to figure 3.5c. On low frequency curves, or if the samples are positioned far apart (along x), the CDE is similar to an antialiased line. When the frequency of the curve increases, or when the curve is sampled multiple times per horizontal pixel, the CDE will resemble a 2D kernel density estimate providing the density of the samples' y position given a x or *time* position. Figure 3.7 shows this

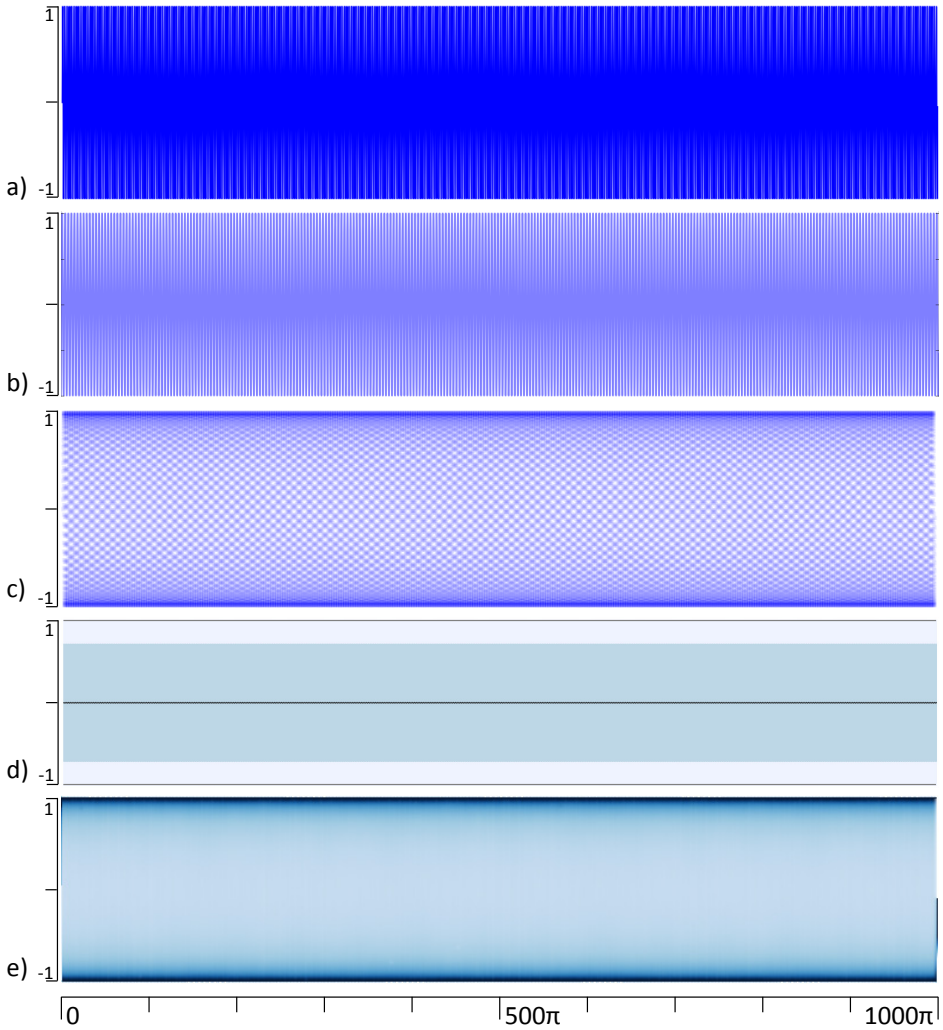


Figure 3.5: Five visualizations displaying the sine curve from zero to 1000π. In the top figure, a, an opaque line is used, and, because of overdraw, only the extent of the function is visible. In the second figure, b, a semi-transparent line is used. The third figure, c, is a scatter-plot of the samples drawn semi-transparently, and it shows the same distribution as the histogram. The fourth figure, d, is aggregated using a moving mean, a moving standard deviation and a moving extent. In the bottom figure, our technique, the Curve Density Estimate, is applied, and the distribution corresponds with that found in the histogram in Figure 3.6.

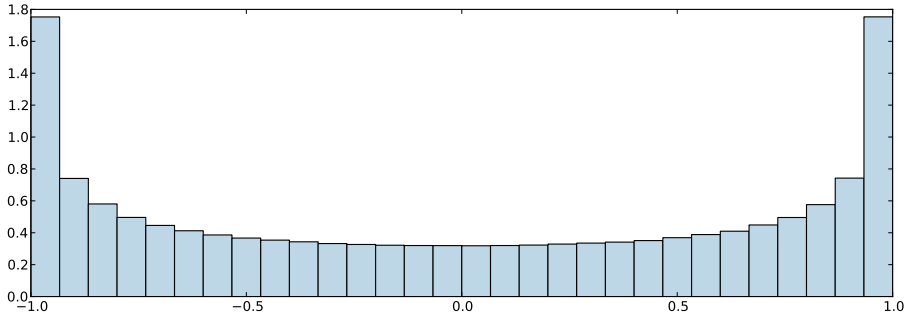


Figure 3.6: 30 bins histogram of $y = \sin(x)$ for regularly sampled values of x.

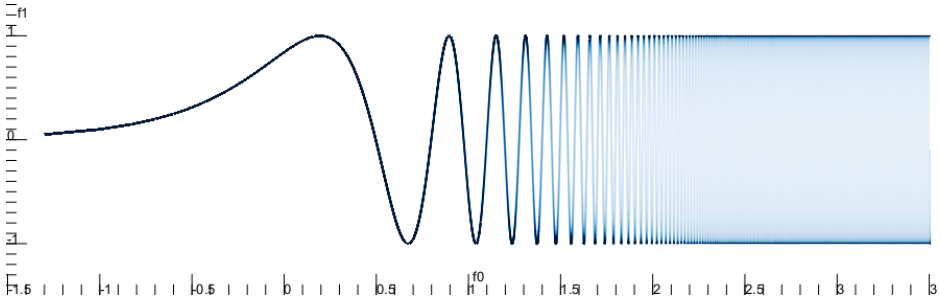


Figure 3.7: A curve density estimate of the sine curve. The x axis uses a logarithmic scale, which shows the smooth transition between a low frequency curve to a high frequency curve to the density estimate of a high frequency curve. The logarithmic exponent is given on the x axis.

smooth transition between a low frequency curve to a high frequency curve by drawing the sine curve on a logarithmic x axis.

2 Interaction with Kernel Density Estimates

This section describes three novel interaction techniques for the interactive visual analysis of process data. The first technique, proposed in paper A, entails an automaticaly updated bandwidth tied to the screen size, instead of, as usual, in terms of the data space. This automatic bandwidth allows the user to zoom in and out while the KDE automatically updates its level of aggregation, resulting in meaningful densities regardless of zoom-level. The second technique, as proposed in paper D, enables a differential analysis by interactively defining difference views. The third, as proposed in paper E, describes a technique on how to interactivly build statistical models that describe parts of the data.

2.1 Screen-Space Bandwidth Automation

Silverman [108] describes kernel density estimates according to equation 3.1 where h is denoted the bandwidth. This bandwidth, also called kernel size, smoothing parameter or window width, has seen several research works suggesting different techniques to identify the optimal bandwidth. Silverman introduced the normal scale rule [108] as a global bandwidth estimator for distributions that are close to a normal distribution. The normal scale rule leads to over-smoothing, however, if the data does not adhere to a normal distribution [130]. In paper A we recognized that the bandwidth can be expressed in screen space, rather than in data space. The reasoning behind this change is to enable interactivity, e.g., zooming and panning. A fixed bandwidth will, depending on zoom level, either be smaller than a pixel, or larger than the entire viewport. The visual effect of a bandwidth smaller than a pixel is that, either nothing is shown, or that a strongly aliased kernel is shown. The visual effect of a bandwidth larger than the display however will result in a near constant value all over the screen.

Figure 3.8 shows an example, where first the entire coast of Norway is shown, and then, by zooming, the city of Stavanger is brought into the viewport. Our automatic bandwidth adjustment sets the bandwidth to approx. 50 km in the first image, ≈ 10 km in the next, and ≈ 1 km in the last. An interesting aspect with this interaction is the seamless and smooth aggregation of all traffic when zooming out again. In relation to the image showing the largest area (the leftmost in figure 3.8), the final zoom (shown on the very right) makes out approximately 20 by 30 pixels, yet all the traffic from the final zoom is aggregated, and contribute to the larger one.

In a right hand system, a viewport is defined by the two points, in data-space, the lower-left $\mathbf{p}_1$, and the upper-right $\mathbf{p}_2$. Furthermore the viewport is defined by the screen-size in pixels $\mathbf{s}$. We define the size of a pixel in data-space as $\mathbf{q} = \frac{(\mathbf{p}_2 - \mathbf{p}_1)}{\mathbf{s}}$. From the previous deduction we showed that the bandwidth $\mathbf{H}$ must be significantly smaller than the viewport, but larger than the size of a pixel, i.e., $k \cdot \mathbf{r} > \mathbf{H} > \mathbf{q}$, for a constant k. The size of this constant k, and its influence on the visualizations, is explored in paper A, but for simplicity, most of the visualizations shown here use a $2 \leq k \leq 20$.

This technique does not make any assumptions about the data distribution. To utilize it to create an overview, the viewport should be set up first so that it covers all of the data and the bandwidth can be set to around five pixels (further details in paper A). When the user, by interaction, either increases or decreases this bandwidth (to either create a crisper or a smoother/aggregated image) it is the ratio towards the pixel-size which is changed, and not in terms of the data space, since this retains the zoom independence.

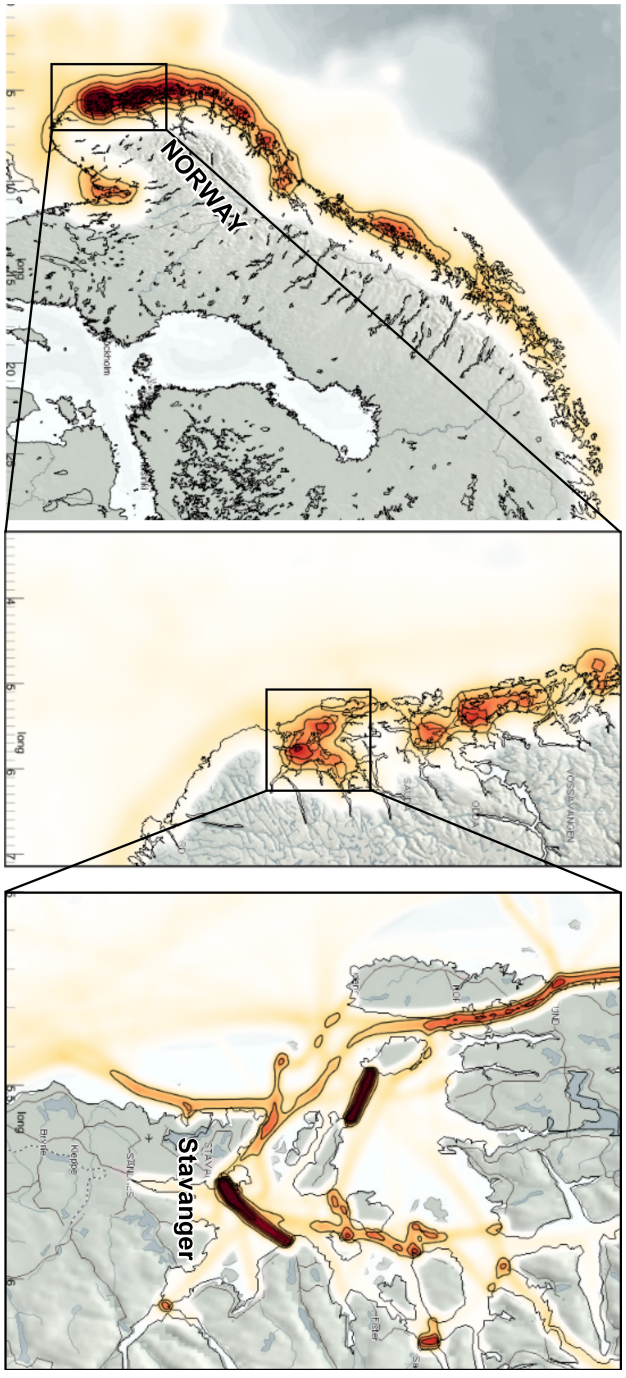


Figure 3.8: A density estimate of non-stationary vessel traffic along the coast of Norway. The bandwidth is automatically updated when zooming in (left to right) towards the city of Stavanger.

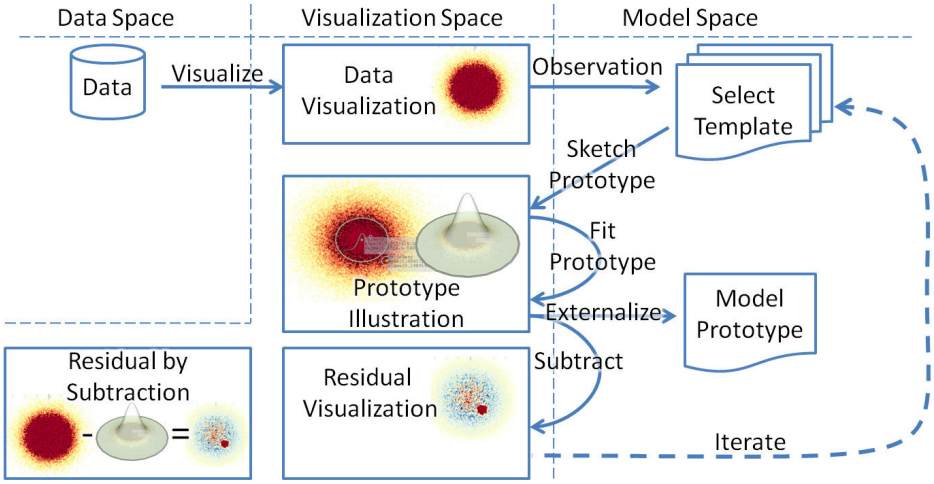


Figure 3.9: Our proposed workflow: visualize and observe, sketch and fit, externalize and subtract, then iterate.

2.2 Model Sketching

In most fields dealing with natural phenomena, e.g., physics or statistics, modeling is an important part of making sense out of data and understanding the underlying structure. Modeling serves multiple purposes, e.g., to develop a basic understanding of how a phenomenon works, prediction, or simplification (parameterization). Modeling is often performed as trial and error, based on both observation and intuition. Modeling is also often considered a global property, such as tests for normality, or calculations of skew. When multiple processes are interacting, it is important to be able to both specify data locally (select a subset) and perform the modeling on this selection. We propose a workflow where the user is enabled to quickly sketch models locally, aided by automated fitting and feedback algorithms. The workflow is detailed in figure 3.9 as *visualize and observe, sketch and fit, externalize and subtract, then iterate.*

As an example for this workflow, consider a simple dataset, as shown in figure 3.10, consisting of two features, one large 2D normal distribution with parameters $\mu = [0, 0]$ and $\sigma = [1, 1]$ and one small artefact represented by a normal distribution scaled by 0.05 with parameters $\mu = [1, 1]$ and $\sigma = [0.2, 0.2]$. The first step of the workflow is to visualize the data and, by observation, decide upon a suitable model. In the middle of figure 3.10 the normal distribution seems like a good candidate model.

The second step of the workflow, *sketch and fit*, starts with interactive sketching. The interface to sketch the normal distribution is to simply click near the observed mean of the data and modify the variance by rolling the mouse wheel. When satisfied with the sketching, the user lets go of the mouse button, and a

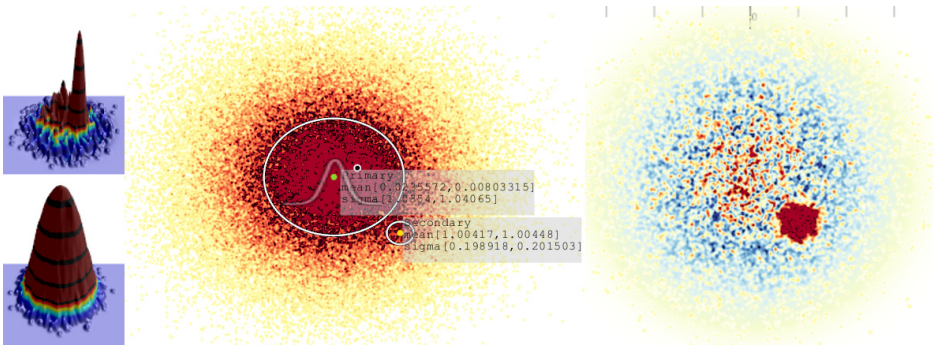


Figure 3.10: Several KDE plots of a normal distribution with a small artifact. The lower left and the middle figures show the (logarithmic) KDE of the data. The images on the right, and the upper left, shows the KDE after abstracting the primary feature and thus clearly revealing the secondary feature (the artifact).

slowly iterated optimization starts automatically. This iteration is intentionally slow to enable the user to stop it if it would diverge from the users intended local optimum. However, if the iteration converges, and remains un-interrupted, iterations of Newton's method is applied until convergence.

The third step of the workflow, *externalize and subtract*, starts when the user accepts the output from the automatic fitting algorithm. The result from the second step and the example, in figure 3.10, is the parameters for the selected normal distribution. In this case the output parameters are: a) the selected model, a normal distribution, b) its mean, $\mu = [.02, .01]$, c), variance, $\sigma = [1.035, 1.04]$, and d) the error measurement sum of squared differences. Together these parameters represent a model instance. This model instance either represent an externalization, where the result is understood, and stored, or represent a "simplification", where the data can be replaced by the models parameters. An important contribution of this workflow is the residual visualization which is made by subtracting the fitted model from the original KDE. This residual is shown to the right of figure 3.10, and since the large feature is removed from view, the smaller feature is clearly visible.

The fourth step of the workflow, *iterate*, starts the procedure over again. The smaller feature which is now visible can be modeled in the same way as the first, and then extracted/externalized, yielding $\mu = [1.004, 1.004]$ and $\sigma = [0.1989, 0.2015]$ vs. the reference $[1, 1]$ and $[0.2, 0.2]$ in this example.

2.3 Differential Analysis

Analyzing differences, or comparative visualization, between two results can either be performed by placing them side by side, or by animation from one to the other [126]. These two techniques provide the images for comparison (either

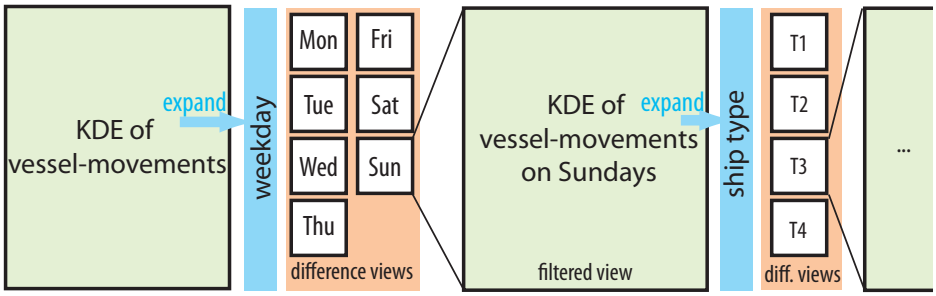


Figure 3.11: Iterative data exploration via difference views.

spatially or temporally), and the extraction of the exact differences is left to the user. When the results are provided as a scalar image/field, however, the differences can be presented utilizing a direct difference operator as one single image called direct comparative visualization by Pagendarm and Post [92]. This image provides exact difference measures and a quantitative result. This exact nature of the difference views are a strength that should be utilized instead of side by side views when possible and desirable. One major problem with the facilitation of difference views into an analysis framework is how to define *what* to take the difference between.

In paper D we propose a workflow on how to specify and analyze difference views. Our basis for a difference view is the KDE plot, as shown in section 1.1, which supports the difference operator. The workflow is described around a concept we call *expand over*.

1. Initialize by creating a KDE plot, which entails selecting two attributes/dimensions and optionally a third attribute as the weight (compare to section 1.1), and view parameters.
2. Select a parameter to *expand over*.
3. If the expand parameter is categorical, the number of categories defines the variable N.
4. If the expand parameter is continuous, an arbitrary[3] number of bins is chosen, e.g., four, six or nine, and this number defines the variable N.
5. Create N views each containing only the samples filtered by either the corresponding category or bin, and subtract this density field from the average[4].

[3]The number of bins is arbitrary in principle. However, often an application-dependent categorization exist, e.g., the Beaufort scale for wind speed, that would lend itself as the basis for the binning. This number should not exceed the practical limitations wrt. screen space and optimally be a $n \cdot m$ multiple.

[4]The average field is defined by all the samples, unfiltered, but normalized

6. The N views show diverging[5] values compared to the average. By inspection the user selects one view.
7. Remove all the introduced views, and replace them with one view, similar to the original one, but with the filter from the selected view applied.
8. Repeat from step 2.

Figure 3.11 details this proposed workflow (paper D) on how to facilitate difference views in a particular visual analysis context.

Paper E also describes differential analysis, but in a different context. When sketching models, both the automatic fitting internally, and the visual feedback on the quality of the fit is provided by differential analysis. In the automatic fitting, the difference between the model, and the data (for the L1 norm), or the squared difference (for the L2 norm) is used internally for the optimization. As a visual feedback, both for determining the quality of the fit, and for residual analysis, the difference is shown where the models are subtracted from the view of the data.

3 Curve-centric Volume Reformation

Up to this point the visualization and interaction techniques described apply first and foremost to the 2D domain. In this section we investigate how we can combine and compare traditional 2D parameterized visualizations in an exact and quantitative way to 3D volumetric data. This comparison is achieved by extracting the volumetric neighborhood of the curve. In this way measurements along this curve are put into a direct context with the volumetric data, in which there might be a correlation. The proposed technique is designed to exploit a logical 1D parameterization within the 3D volume, e.g., a sensor moving in a volumetric space. The sensor collects data and its movement will define a curve in space that has a 1D parameterization either in time or in terms of arc-length.

On the left of figure 3.12 a synthetic volume with a curve within it is shown. We first consider this volume to be geology, and the curve as a well. The reformed volume, to the right of figure 3.12, contains only the neighborhood of the well. This extracted neighborhood can influence either the measurements or drill procedure and is therefore often an important context to provide.

The general idea is to reform (or deform[6]) the volumetric data in such a way that the 1D parameter-space from the curve within the volume is aligned and parallel with one of the two dimensions on the 2D screen. This alignment is then used to compare across multiple modalities of data. Figure 3.13 shows such a

[5] A color scheme that emphasizes the extremes on both sides of a middle range, or here positive and negative values, compare to Brewer [54]

[6] Definition by Merriam-Webster: Reform – to put or change into an improved form or condition. Deform – to spoil the form of or to alter the shape of by stress

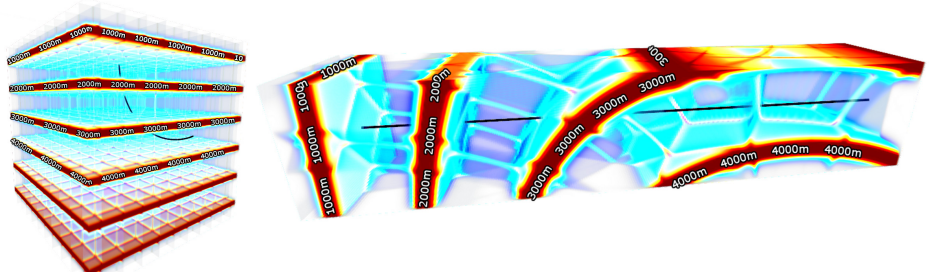


Figure 3.12: Left, a test volume with a curve in it, and right the result of Curve-Centric reformation. The curve is, after the reformation, the straight line shown in the middle of the volume to the right.

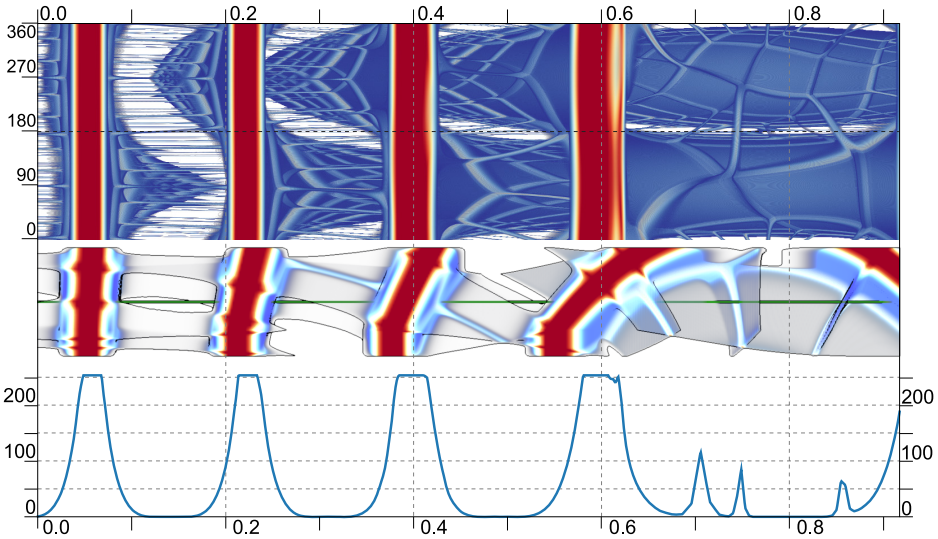


Figure 3.13: Three different techniques, applied to the same data: on top a reformed volume, in the middle radial raycasting, and below, sampled values from the volume as a function graph. This comparative visualization allows the accurate comparison of intensity values to their spatial neighborhood.

comparison where three different views on the same data highlight the vertical alignment.

In order to perform a curve centric volume reformation we first define a moving coordinate frame. The moving frame is a local coordinate system, or a tensor containing orthonormal vectors for every point on a curve $\mathbf{r}(t)$. In our work we provide a moving coordinate frame that is both smooth and that adheres to a

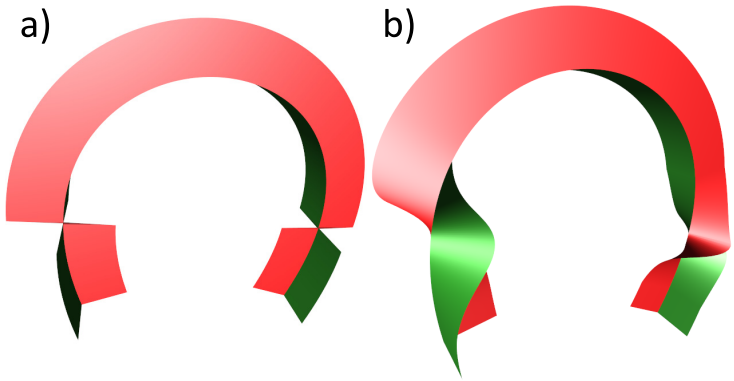


Figure 3.14: Surfaces following the normal $\mathbf{N}(t)$ as red, and the binormal $\mathbf{B}(t)$ as green. Figure a) shows the modified Frenet frame, experiencing a sign change that our smoothed version, b), handles gracefully.

user specified up direction. This moving frame is similar to a Frenet frame [43] but it does not require the second derivative, and we apply smoothing using quaternions to the tensor. This smoothing accomplishes two important features. The first feature is the solution to the problem where the Frenet frame collapses and potentially switches sign at the point of inflection. The second feature occurs when dealing with either noisy or high frequency movements, which without smoothing will lead to an overly distorted neighborhood. As an example for the first solution, where smoothing works around discontinuity problems at a point of inflection, two sample moving frames are shown in figure 3.14. In this figure the Frenet frame is compared to our technique and two of the three orthonormal vectors are shown using a colored surface. Note that the discontinuities at the points of inflection are removed in our coordinate frame. The second issue, caused by enforcing the tangent vector of the curve into the tensor, is also solved by loosening this definition by smoothing. Figure 3.15 shows how different results are given in two different cases, one where the tangent is smoothed/kept constant, and the other where the tangent vector is equal to the derivative of the curve.

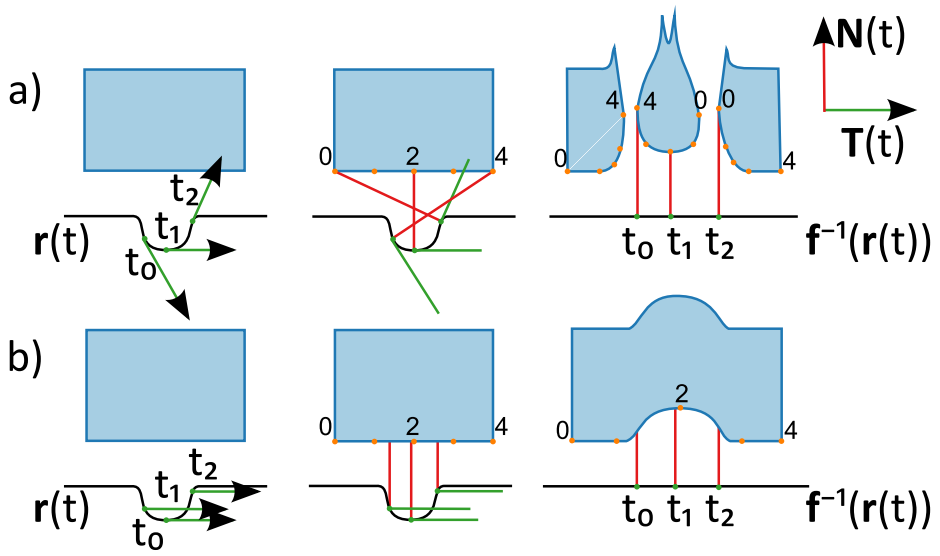


Figure 3.15: TThe blue box represents a volume, and the black line the curve to straighten. Two rows showing a) curve-linear reformation with tangential $\mathbf{T}(t)$ and b) the same curve-linear reformation, but here with constant tangents. Columns left to right show, tangents, normals, and lastly the deformed box.

Chapter 4

Demonstration

This chapter demonstrates the applicability of the techniques as described in chapter 3. It is divided according to the two industries that we collaborated most with, i.e., the petroleum industry, and maritime vessel traffic monitoring.

1 Drilling for Oil and Gas

This section provides three examples. Two exemplify the interactive visual analysis of streaming data using kernel density estimation. The third addresses the positional uncertainty associated with drilling.

1.1 Streaming Drilling Data

Streaming data, as introduced in section 2, are made from data-samples arriving in a sequential manner. Two important aspects of streaming data visualization were, first, the unknown *dataset size*, and second, that the stream in its entirety cannot be stored, only viewed *in situ* with a limited history. In the case of data from drilling operations, however, all the data is usually stored in massive databases. Still, for visualization and monitoring applications with limited resources, the practical scenario is that of an in situ setting.

The most prominent visualization of streaming data in the industry today is that of windowed time graphs, showing anything from the last hour to several hours. Such a graph, a typical drilling data visualization, or a dashboard, is shown in figure 4.1. In a typical operation center this would be placed on a large wall-display and connected with one or several drilling operations. This dashboard would then be used as a "background" to get a glimpse of the current status of the running operations, and a more thorough investigation would be done through other tools.

Progress Overview

Where understanding the recent history with the use of windowed data visualizations is common, getting a larger overview of longer processes / time-series is often problematic. In paper A we introduced a visualization where streaming

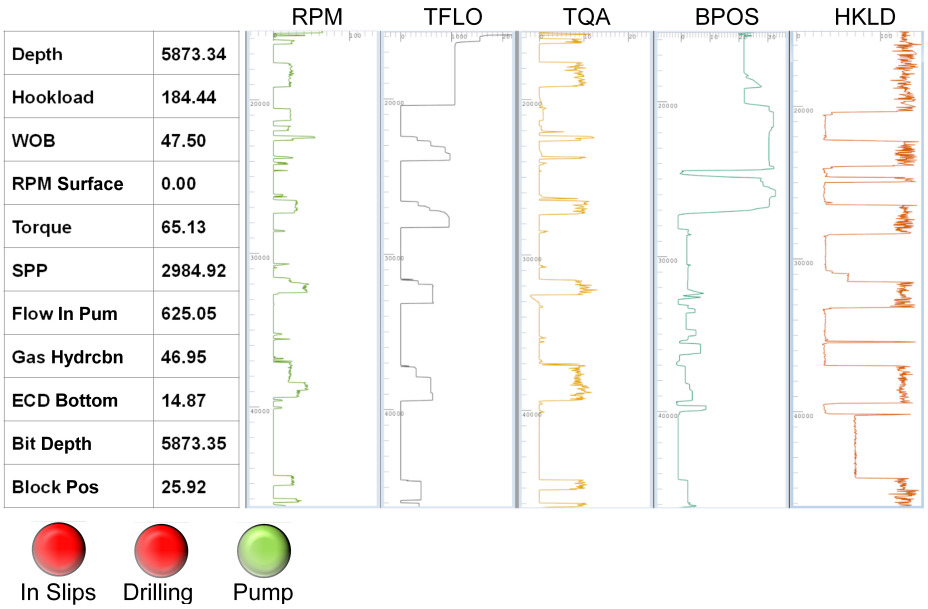


Figure 4.1: A typical dashboard visualization showing the recent history of a drilling operation with one graph per measured attribute (time displayed vertically) and the most recent values on the bottom. This figure also displays the exact values and some derived binary status indicators.

data are added to a kernel density estimation, eliminating the storage problem of the stream. Figure 4.2 shows streaming data added to a kernel density estimate showing the "time spent" distribution over *hook-load* and *measured depth*, two common parameters to visualize the progress in drilling operations. Using this visualization the user can observe where the majority of time has been spent in the drilling operation. Furthermore, since this visualization is quantitative, it enables user to mark an area and have the sum (or integral) interactively, and instantly, calculated, and display its result in (the unit) minutes; also shown in figure 4.2. This particular drilling operation, as shown in figure 4.2, stalled at just over a thousand feet, which was indicated by the large density there. The user specified box contains 76 minutes of non-productive time, which would need to be accounted for by further inspection or inquiry.

Investigating Friction

In paper E we introduced an additional interaction technique to KDE where the data can be modeled through interaction, without access to the data (as a requirement of the streaming access). A task often performed during drilling operations

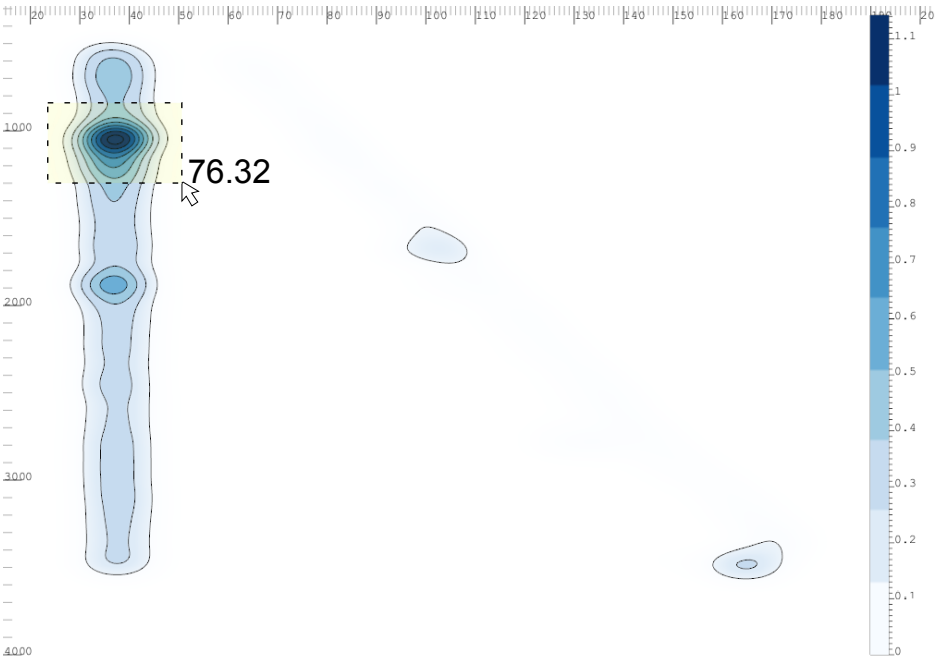


Figure 4.2: A kernel density estimate showing the distribution of time in a drilling operation. Highlighted in the top left is a large buildup of nonproductive time, which would need further inspection. Here, a large bandwidth (to smooth the data) is used to de-emphasize the details, leaving a high level overview.

is that of creating a *road map*. The road map is a chart of the expected/simulated frictions that will be encountered on the way down. The friction is an important parameter to monitor since it is an indicator of several potential problems. E.g., when there is a buildup of cuttings (loose materials), this can, in the worst case, lead to a stuck pipe. More frequently, however, but also potentially dangerously, this can lead to a *pack off*. A pack off occurs when the buildup of material gathers behind the wellbore and plugs the hole around the drill string. This plug leads to an increase in pressure, due to the loss of circulation. This pressure spike will often fracture into the formation, and lead to loss of mud, or influx.

The problem is, however, that friction cannot be measured directly, but needs to be derived from either the modified weight during movement, or derived from the measured torque while rotating. A common operation to perform at certain depths is called a *rotation off bottom* or a ROB. The drill bit is lifted off the bottom and a given rotation or RPM is maintained and the torque is measured. A road map will contain the expected torques on these given depths and is used to compare against these measured torques. Figure 4.4 displays the KDE of torque over depth, and is the result of accumulated streaming data. Using our interactive

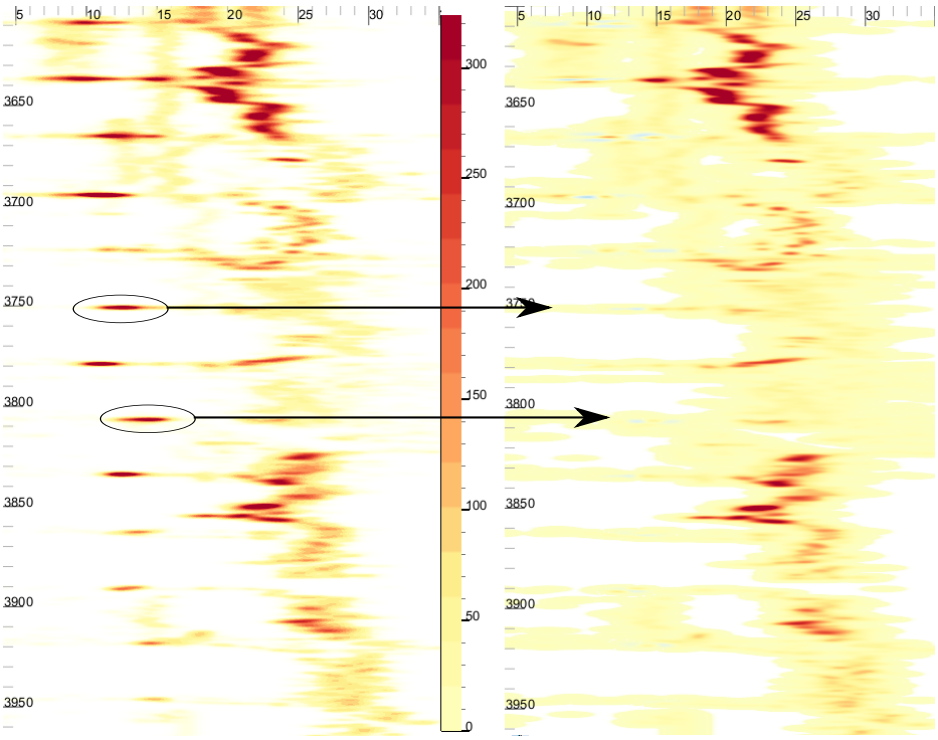


Figure 4.3: Torque in kN.m over depth. The figure on the left shows the original data, containing some ROB tests, which then were modeled and subtracted leading to the residual view on the right.

model building techniques the user sketches each rotation off bottom procedure, and extracts the depth and torque along with their respective variance. These extracted values can then be inserted into the roadmap, as an abstracted model of friction shown in figure 4.4. This figure can be compared against the predicted friction, and the calculated error bars can be used to determine the risk within given probability thresholds.

1.2 Positional Uncertainty

There are several uncertainties associated to the process of drilling. Such uncertainties can result from individual measurements where an error might be included, but these are often well understood and modeled. The larger uncertainties are both in what exactly one will find in the subsurface, but also in understanding exactly where the position of the drill bit is. The positional uncertainty is a cumulative error that can be modeled fairly well. The calculated potential position of the drill bit, at any given time, results in an ellipsoid-shaped

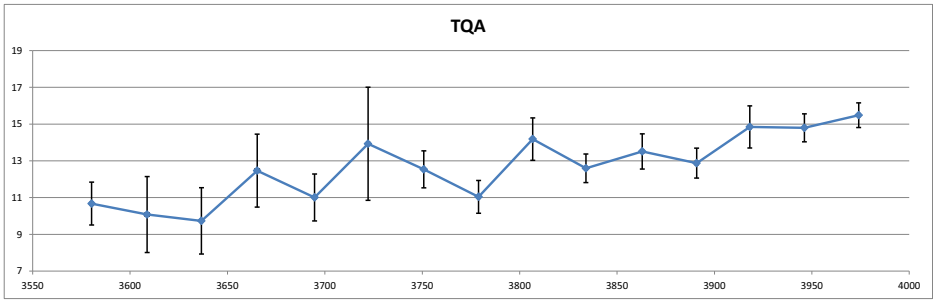


Figure 4.4: Changing torque in kN.m over depth in feet, for a series of ROB tests. The abstracted results from Fig.4.3 are shown in a graph with error bars at 1σ for both depth and torque uncertainty. In this figure, the uncertainty bars for depth are so small they cannot be seen, and are thus negligible.

field of uncertainty, with its mode the highest likely position. In figure 4.5 the ellipsoid, or ellipse in 2D, for 95.4% positional certainty is shown. An important part of the positional uncertainty lies in determining where the drill-string crosses over from one formation to the next. If there is a pressure difference within these, such as often found when entering a reservoir, special precautions must be made. These precautions are needed to make sure that the pressure inside the well does not drop below the pressure in the formation, at the same time as it should not rise above the formation fracture pressure[1]. One such precaution is the setting of a new casing[2], which allows adjusting to a higher pressure further down while protecting the formation above from fracturing.

The predominant way to observe the progress of the drilling process is measured in feet along the *measured length* or arc length, and to compare this to the planned well path. This one-dimensional parameterization of the planned path is shown along with the expected stratigraphy (expected layering) and the planned casings or milestones as a drill plan and overview. While the simplicity of this design serves its purpose well, there are several problems which could be addressed. The first problem is that this 1D stratigraphy is only correct given that the planned path is followed exactly, and the second problem is that the three-dimensional position error is reduced to a one-dimensional measure. In figure 4.5 the positional error is shown both in three dimensions, and in the described one dimensional case. Relying on the 1D stratigraphy combined with the projected error ellipse, this visualization will not indicate that the next formation is possibly entered. However, in the three-dimensional view, the ellipse is touch-

[1] Pressure above which the injection of fluids will cause the rock formation to fracture hydraulically.

[2] A casing is the insertion of a metal pipe almost as large as the well hole. Cement is then used to fill the gap outside the casing, permanently protecting the well from the outside formation.

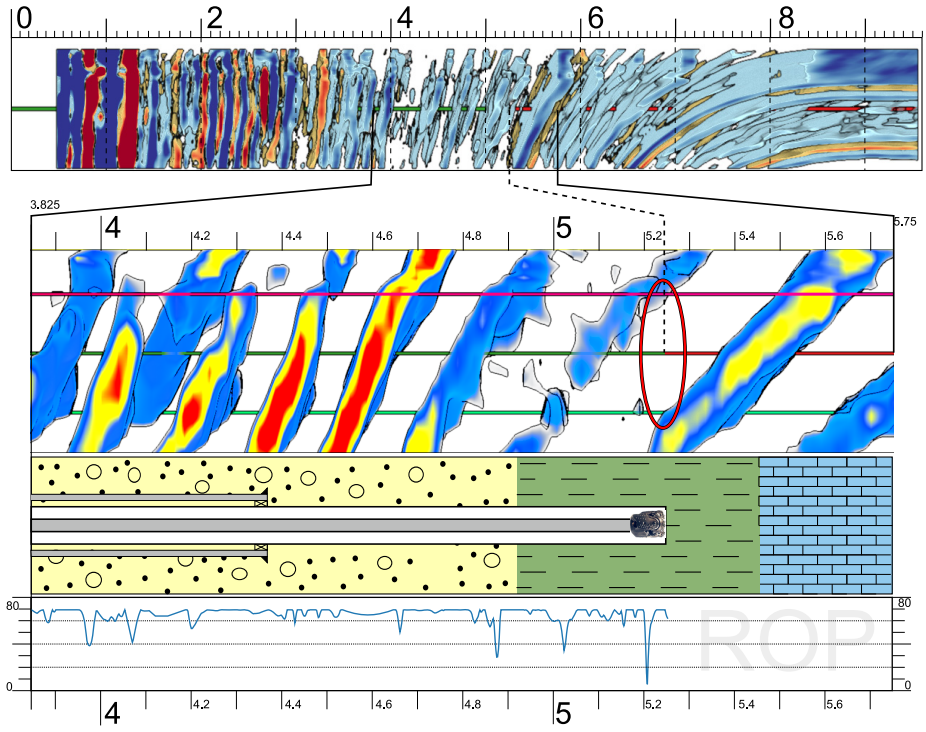


Figure 4.5: A reformed 3D seismic data visualization can provide spatial reference for real time drilling as well as showing uncertainty in the 1D lithology column (the second from the bottom image). The image on top shows the full length reformed wellbore, below a zoom-in of the section currently drilled, which also contains the 1D lithology/stratigraphy with the current drill bit position, and a real time graph showing the rate of penetration (ROP) for the section. The red ellipse in the center shows the positional uncertainty.

ing the next formation, and we can thus not guarantee with a 95.4% probability safety margin that we are still outside. This could provide an alert, cautioning the operator to be wary of other signs or indications of the increased pressure.

2 Differential Analysis of AIS Data

The Automatic Identification System, AIS, has the primary function of supporting collision avoidance, so that two vessels can detect each other's signals, alert and take corrective actions. In Norway the coastal administration has a series of receivers along the coast which collect and store the movement data of all the vessels within range. We were contacted by the Norwegian Coastal Administration

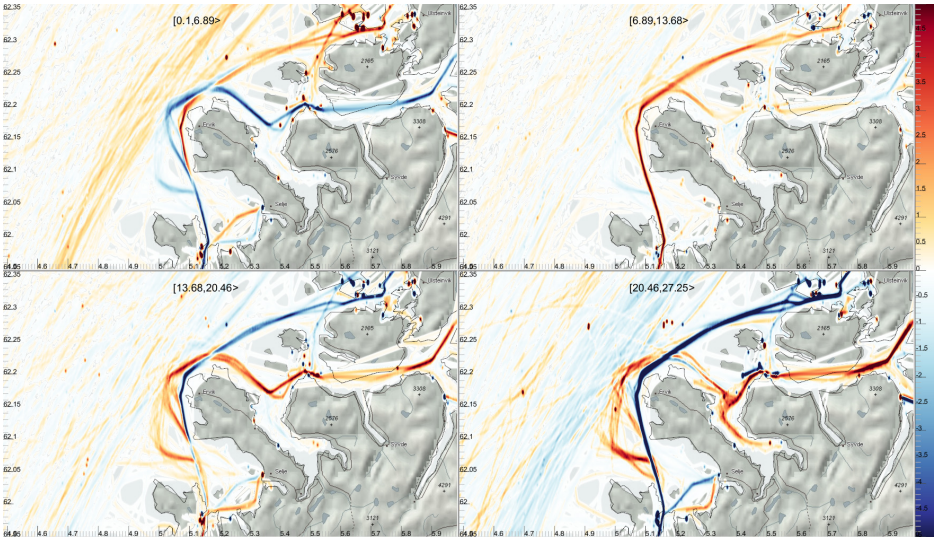


Figure 4.6: Passing vessels outside the Stad peninsula, and their changed movement pattern given stronger winds. From lower winds to stronger winds from the upper left to the lower right. Red colors indicate more than average traffic in that interval of winds, and blue colors indicate less than average.

to analyze AIS data related to questions on a proposed sea tunnel through the peninsula of Stad (detailed in paper D). One of the questions was to investigate how the traffic pattern around Stad changes due to the varying weather. Using the workflow proposed in paper D we first created a KDE of all the traffic around Stad. We then *expanded* this view with the wind-speed attribute, resulting in the visualization shown in figure 4.6. Our expand technique first categorized wind speed into four bins, then visualized the traffic given one wind speed bin subtracted by the average traffic. Investigating these four views (in figure 4.6) reveals that the red paths (more traffic than average) move further out from the coast outside Stad given stronger wind speeds. This indicates that the vessels opt for the safer route as wind speeds increase, something also shown north of Stad, where more than average vessels pull closer to the coast. This visualization (and other statistics presented in paper D) was handed off to the NCA which will potentially use them in their final recommendation to the government.

Chapter 5

Conclusion and Future Work

A starting point for this PhD project was to address the visualization challenges associated with the introduction of, and a possibly more widespread adoption of, technology enabling more advanced sensors during the drilling for oil and gas. We specifically sought to target the wired pipe [16] technology. A technology that increases the potential communication bandwidth with down-hole sensors by several orders of magnitude. This initial problem definition is quite narrow in terms of the application domain, but the visualization techniques that could be used to address it were open and left up to the researcher. From early on we recognized the strengths in creating solutions which were applicable beyond the application domain and to generalize our visualization techniques. We are confident that we have proven, through the application examples given, that we have achieved this.

An interesting research topic that we found highly applicable to process and streaming data was that of KDE based visualization. From our experiences, as documented through our publications, we see the biggest potential applications and usages for KDE-based visualizations being:

- As a frequency view, for dealing with occlusion, clutter, and thus scale beyond the number of samples a scatter-plot can coherently display.
- As a fixed memory representation for large streaming datasets. Independently of stream-size the memory usage of the KDE-based visualization remains constant.
- As a normative and quantitative visualization for samples with a unit that provides meaningful aggregates.
- As a visualization that supports algebraic operations, e.g., difference views, or, divide the KDE, of total contributions to a campaign per square mile, with the density estimate of people per square mile and get an estimate of contribution per person over an area.
- As a scale independent visualization technique, either through zooming, or through the usage of multiple bandwidths.

In paper E we introduced a technique that enables interactive visual analysis on this fixed memory footprint representation. As far as we know, this visual analysis on visual representations (as abstractions for the data) is an unexplored area of research, and is also an area we would like to see explored. Through our research

we addressed several challenges that we encountered during the exploration of KDE in visualization, but several more still exist. In future work we would like to see several challenges related to KDE-based visualizations addressed, including:

- Further explore the visual analysis of visual representations as an intermediary to the data.
- Using the same bandwidth for all samples is often not suitable. E.g., using the same bandwidth for two distinct distribution clusters. A variable kernel density estimate, however, sets an optimal bandwidth per sample, based on its local distribution.
- Exploiting the fixed memory bandwidth of KDE in large streaming data requires additional techniques on how to combine this with zooming and panning, e.g., rendering to a tile based map solution.
- The curve density estimate provides a continuous representation of the distribution of the curve. Modeling this distribution (e.g., detecting modes) could abstract the curve in ways that a moving average could not.

The implementation done for the different papers in this PhD project have been combined into a framework for interactive visual analysis, named Enlighten, as described in paper D. In the later years this application has also seen commercial usage in consultancy for different domains, additionally strengthening this research's applicability.

Part II

Scientific Results

Paper A

Interactive Visualization of Streaming Data with Kernel Density Estimation

Ove Daae Lampe[1,2], and Helwig Hauser[1]

[1]Department of Informatics, University of Bergen, Norway
[2]Christian Michelsen Research, Norway

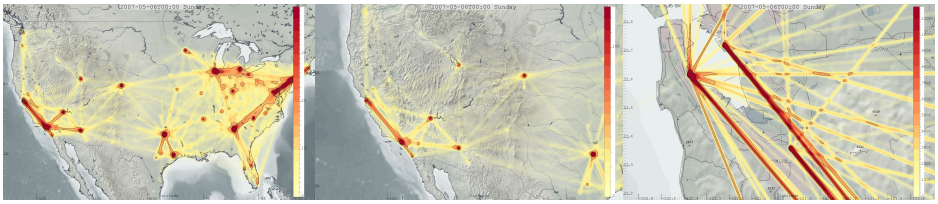


Figure 1: Interactive zooming towards SF Bay, where at first all the traffic from the Bay Area is aggregated, to a view where we can separate traffic from the three major airports, and even the distribution of traffic in each airports' cardinal direction. This interaction is enabled by automatically updating the bandwidth of the KDE when the viewport changes.

Abstract

In this paper, we discuss the extension and integration of the statistical concept of Kernel Density Estimation (KDE) in a scatterplot-like visualization for dynamic data at interactive rates. We present a line kernel for representing streaming data, we discuss how the concept of KDE can be adapted to enable a continuous representation of the distribution of a dependent variable of a 2D domain. We propose to automatically adapt the kernel bandwith of KDE to the viewport settings, in an interactive visualization environment that allows

This article was published in *Proceedings of the IEEE Pacific Visualization Symposium 2011*, pages 171–178, March 1–4, 2011 and presented at PacificVis in Hong Kong by Ove Daae Lampe.

zooming and panning. We also present a GPU-based realization of KDE that leads to interactive frame rates, even for comparably large datasets. Finally, we demonstrate the usefulness of our approach in the context of three application scenarios – one studying streaming ship traffic data, another one from the oil & gas domain, where process data from the operation of an oil rig is streaming in to an on-shore operational center, and a third one studying commercial air traffic in the US spanning 1987 to 2008.

1 Introduction

The scatterplot is one of the most prominent success stories in statistics and visualization. Scientists and practitioners have used scatterplots for more than 100 years to study the distributional characteristics of multivariate data with respect to two data attributes or dimensions [119]. However when datasets are large, scatterplots are challenged by overdraw and cluttering. There are approaches to improve this situation, e.g., by employing semi-transparency during rendering or by subsetting prior to the visualization [34]. With such approaches the number of data items that can be effectively shown in a scatterplot can be pushed by one or two orders of magnitude. Beyond a certain point, however, at least when there are many more data items to be shown than there are pixels in the scatterplot, the item-based approach is collapsing [90]. It has been shown that switching to a frequency-based visualization metaphor is a useful solution in such a case [90, 39, 87, 136]. Such frequency based visualizations are e.g., histograms or density estimations.

While histograms are straightforward to implement and interpret, the parameterization of data introduce a significant variance in appearance, e.g.,the discretization of data into buckets/bins, may cause aliasing effects. Corresponding interpretations depend on bin count and interval range along the axis. [116]. Fig. 2 illustrates one example of such a major change by showing two histograms of the same data – one computed with 9 bins and the other one with 10. To achieve a more truthful assessment of distributional data characteristics, Kernel Density Estimation (KDE) [108] is commonly used in statistics. Assuming that the distribution of the data items adheres to a certain probability density function (PDF), KDE allows estimating this PDF from the samples. The result is a function that represents the distribution of the data items in terms of their density in the data space. Years of research has made KDE into an important tool for statistical data analysis [130]. One of the major advantages of KDE is that it directly evaluates the data, without imposing a model onto it, which, consequently has the advantage that the data speak for themselves. (as Silverman says [108]).

Our goal of using interactive visual analysis on large amounts of dynamic and streaming data, demanded a real-time KDE implementation. Fast update rates for KDE is needed to highlight the coherency of the temporal correlations. To support continuously updates of streaming data, rules out techniques relying on pre-processing.

With this paper we follow up on this opportunity in utilizing KDE for visualization, and in the following: We propose a line kernel for the KDE-based visualization of streaming data, and an automatic adaptation of the bandwidth used for KDE, according to the zoom level of the visualization. We present a KDE-based interactive visualization, with real-time performance enabled by the GPU. We demonstrate how to visualize the distributional characteristics of an-

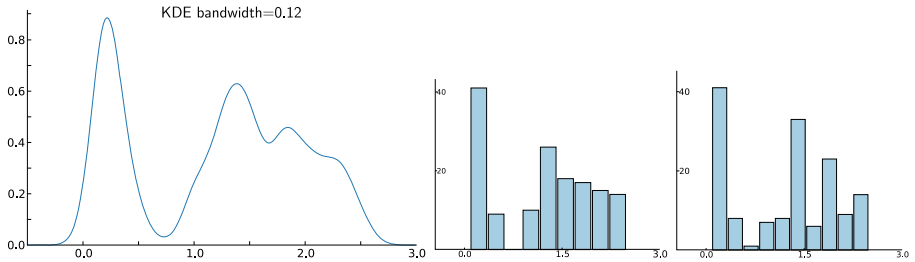


Figure 2: A kernel density estimation of *petal width* in the Iris dataset [40] and two corresponding histograms, one with 9 bins and the other with 10 bins.

other data attribute (instead of sample frequency) by adapting KDE accordingly. The usefulness of this approach is showed in three demonstrations, one on surveillance data from the maritime traffic domain, one on real-time drilling data from the petroleum industry, and one on air traffic data.

2 Related Work

Both scatterplots and histograms have become a commodity in data visualization, even for the general mass market. Ericson (from the New York Times) even said that the scatterplot is the most complex visualization technique that the general public can appreciate [35].

An interesting subset of previous work, which also is of special relevance for our work here, comes in the form of examples for this methodological change from an item-based visualization approach (as the classical scatterplot) to a frequency-based approach (such as the histogram). Fisher, for example, visualizes aggregated numbers of downloads with the hotmap approach [39]. Novotný and Hauser demonstrate how the transition to a frequency-based methodology can enable the visualization of very large datasets in parallel coordinates [90] and Muigg et al. show how this transition enables the visualization of hundreds of thousands of function graph curves [87]. Artero et al., who also use a frequency-based approach to visualizing large datasets with parallel coordinates [6], refer to kernel density estimation as an approach to compute the density values (but eventually revert to a box function as their reconstruction kernel). Kidwell et al. refer to KDE for reconstructing a smooth and space-filling heatmap visualization of a small number of data items [73]. For a more thorough discussion on the use of KDE in visualization we refer to the work by Scott [104]. Whittaker and Scott presented the use of the Average Shifted Histogram (ASH) i.e., an alternate and very efficient density estimation, that approximates KDE, for the use in a geographical context [135].

Very interesting related work is an approach called *continuous scatterplots* by Bachthaler et al. [9]. Assuming data that are continuous with respect to a spatial reference domain – such as the distribution of physical or chemical quantities over a 2D or 3D reference space as acquired through measurements or numerical simulation – a mapping is computed that represents the data in the form of an m-dimensional continuous histogram. KDE-based visualization, as discussed in this paper, is not a mapping from a continuous spatial domain, but rather a mapping of sparesly sampled data, which is mapped into a spatial domain. Similar work on the reconstruction of uniformly sampled data is done by Crawfis and Max [26], where they investigate the use of texture splats with normal distributed values, as means to reconstruct the continuous data field in 3D. Similarly, as in the work by Bachthaler et al. [9], a requirement for this technique is the continuous spatial domain. The work presented here also supports these continuous domains and also extends to support streaming time-dependent data, attribute reconstruction, and non-uniformly sampled data.

Jang et al. investigated the representation of non-uniform, tetrahedral volumetric datasets, by weighted Radial Basis Functions (RBF) [61, 21]. They introduce an algorithm on how to effectively render such 3D RBFs by applying a slice based technique. In this work, we investigate the use of a broader category of kernels than those available as RBFs, namely the product kernel and our extended line kernel. We furthermore show that when applying kernels to dimensions with different units or of different scale, RBFs are impractical, e.g., when plotting meters over tonnes.

Andrienko and Andrienko defined a generalized method on how to create abstractions from geospatial movement data [5]. This abstraction technique generates, from unstructured and unrestricted movement data, potential nodes, where traffic can be aggregated, similar to a node-link diagram. While, theirs and our technique both share the same type of source data, the end result portray two different images, with similar, but still, different usages. The result by Andrienko and Andrienko [5] show the total volume of traffic, and how this volume is distributed, i.e., by counting all passing vessels. With our technique we display, where the traffic spend its time, e.g., if a car stops, it will still contribute a kernel at that position. The differences in these two techniques, as well as other techniques that employ node-link diagrams for aggregation, are comparable to that of the histogram on one side, and KDE on the other side. While the aggregation techniques, similar to the histogram, provides a high level of abstraction, and clear benefits in terms of quantitative readouts, they will potentially suffer aliasing effects and hide underlying details which only a continuous representation can show.

3 Kernel Density Estimation

In the following, we first briefly define kernel density estimation (KDE), before we discuss KDE-based visualization.

KDE is a well-proven approach to achieve a non-parametric estimation of data density that has been introduced to the field of statistics by Rosenblatt and Parzen about 50 years ago [101, 93]. Given a set of n (1D) data samples x_i, $1 \leq i \leq n$, the kernel density estimator $\widehat{f}_h(x)$ is computed as

$$\widehat{f}_h(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n} \sum_{i=1}^{n} K_h(x - x_i), \tag{1}$$

based on a kernel function K and a bandwidth parameter h. Often symmetric kernels are considered as K, with $K(x) \geq 0$ and $\int K(x)\,\mathrm{d}x = 1$, also often centered around 0. In such a case also $\widehat{f}_h(x)$ is also nonnegative and integrates to 1. This enables interpretation of $\widehat{f}_h(x)$ as a density function that approximates the PDF $f(x)$ of the data items x_i from which it has been constructed. The KDE of data attribute *petal width* in the Iris dataset, as shown in Fig. 2 on the left, is considered to be a more truthful visualization of the distribution of the considered data values, than a histogram.

A large variety of kernels has been studied, including the uniform kernel (based on the normal, Gaussian distribution), the triangle kernel, the Epanechnikov kernel [124], and many others. In many cases, however, the normal kernel,

$$K(x) = \frac{1}{\sqrt{2\pi}}\, e^{-x^2/2}, \tag{2}$$

is used in KDE. Even though it has been concluded [130] that variations in the choice of K are less important than variations of h, there still are strong arguments for choosing the normal kernel [86], e.g., when calculating the modes of $\hat{f}_h$.

Bandwidth h is a parameter which influences the smoothness of the density reconstruction. Fig. 3 shows four results from a 2D KDE with increasing values of h. Several authors have worked [124, 130] on (automatically) optimizing the choice of bandwidth h, e.g., Silverman describes the normal scale rule [108] to derive an optimal value for h as

$$h := 1.06 \cdot \sigma \cdot n^{-\frac{1}{5}}. \tag{3}$$

This rule is leading to an optimal estimation if the data is normal distributed. It will lead to an over-smoothed result, however, if not [130]. There are also several other approaches to globally optimize h (several covered by Wand and Jones [130]), and in Sec. 6 we briefly discuss why these are not sufficient for interactive visualization, and propose a new approach.

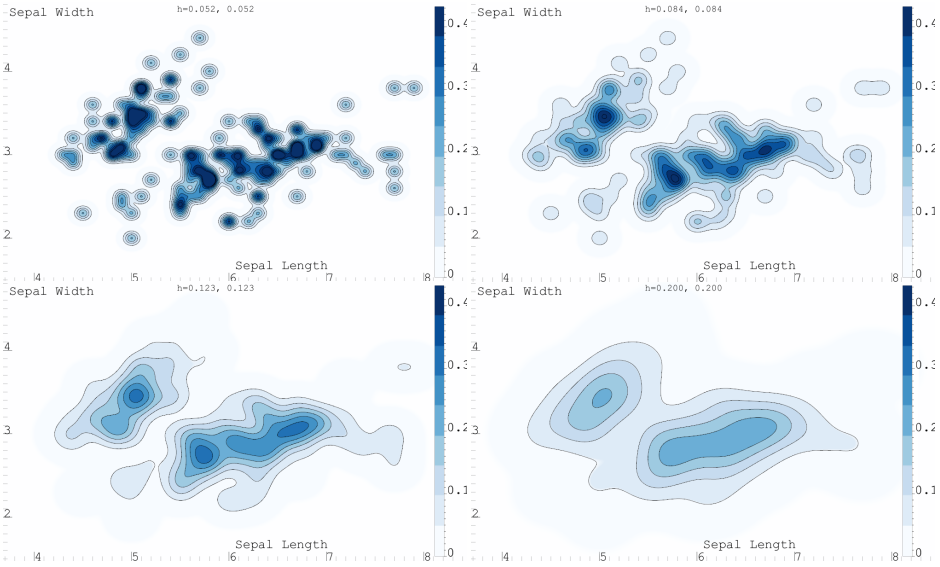


Figure 3: 2D KDE for the Iris dataset [40] with increasing bandwidth.

Altogether, it is generally agreed that KDE is a very appealing tool to investigate the distributional characteristics of data. Gray and Moore write *"In general, density estimation provides a classical basis across statistics for virtually any kind of data analysis, including clustering, classification, regression, time series analysis, active learning, ..."* [48].

Up to here, we have discussed KDE in the one-dimensional case. It is straightforward, however, to extend KDE to multiple dimensions [104]:

$$\widehat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) \tag{4}$$

with $\mathbf{H}$ being a symmetric and positive definite bandwidth matrix and $K_{\mathbf{H}}$ being defined as

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}} \mathcal{K}(\mathbf{H}^{-\frac{1}{2}}\mathbf{x}).$$

$\mathcal{K}$ is a multi-variate kernel function that integrates to 1. For the 2D case – central to all of the following –, we will consider the following simplified form of the bandwidth matrix

$$\mathbf{H}_{2\mathrm{D}} = \begin{vmatrix} h_{1,1} & 0 \\ 0 & h_{2,2} \end{vmatrix}$$

that leads to the following form of a 2D KDE:

$$\widehat{f}_{2\mathrm{D}}(x,y) = \frac{1}{nh_{1,1}h_{2,2}} \sum_{i=1}^{n} \mathcal{K}\left(\frac{(x-x_i)}{h_{1,1}}, \frac{(y-y_i)}{h_{2,2}}\right)$$

Also in 2D, the kernel function $\mathcal{K}$ is usually chosen to be a probability density function. There are two common techniques for generating a multivariate kernel $\mathcal{K}$ from a symmetric univariate reference kernel K [130]:

$$\mathcal{K}^{\mathrm{P}}(\mathbf{x}) = \prod_{i=1}^{d} K(x_i) \quad \text{and} \quad \mathcal{K}^{\mathrm{S}}(\mathbf{x}) = K(|\mathbf{x}|)/c_{k,d}$$

where $c_{k,d} = \int K(|\mathbf{x}|)\,\mathrm{d}\mathbf{x}$. $\mathcal{K}^{\mathrm{P}}$ is known as the product kernel and $\mathcal{K}^{\mathrm{S}}$ as the radially symmetric isotropic kernel. The latter of these kernels is a radial basis function (RBF), and should only be used when a single bandwidth can be devised for all the plotted dimensions. When plotting two different units, or attributes of different scale, we choose the product kernel with individual bandwidth values for the two represented data dimensions. We have now defined kernel density estimation, especially also in its 2D form, and we have compared KDE-based 2D visualization with scatterplots. Later in this paper, as a technical contribution, we present an approach to compute KDEs on the GPU, achieving a speed-up factor of about 100 (compared to existing KDE algorithms), and thereby enabling interactive frame rates needed for this visual data exploration and analysis; even for large datasets.

4 Reconstructing the Distribution of a Third Attribute

In the following we discuss an extension of the KDE concept that allows the visualization of the distribution of a third data attribute (with respect to two other data attributes as in the scatterplot).

We first forgo the normalization in Eq. 4, i.e., we omit the division by the number of data items n, and thereby achieve an estimate function that will integrate to n, accordingly. Next, we introduce a weighting factor c_i to each of the accumulated kernels, that we make dependent on a third data attribute $d_{i,c}$. The new estimate is then defined as

$$\widehat{g}_{\mathbf{H}}(\mathbf{x}) = \sum_{i=1}^{n} c_i K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i) \tag{5}$$

Visualizing $\widehat{g}_{\mathbf{H}}(\mathbf{x})$, e.g., as a height field over the 2D domain of $\mathbf{x}$, will (as a whole) communicate the accumulated sum of all values c_i of data dimension $d_{i,c}$ since

$$\int \widehat{g}_{\mathbf{H}}(\mathbf{x})\,\mathrm{d}\mathbf{x} = \sum_{i=1}^{n} c_i\,. \tag{6}$$

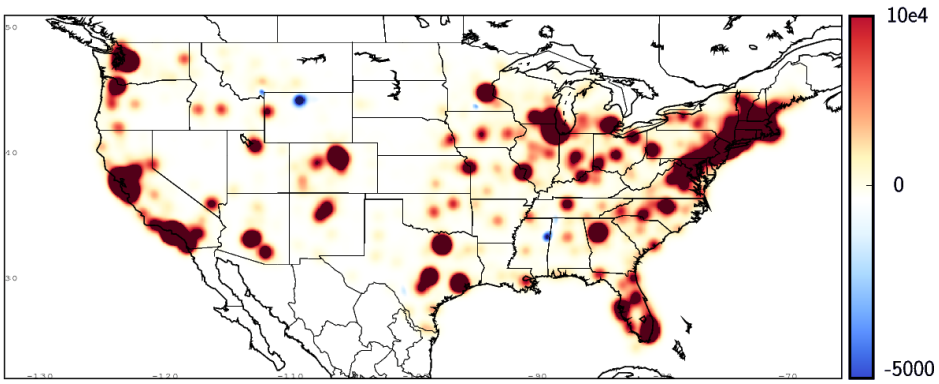


Figure 4: Visualizing over 165 000 monetary contributions to the Obama campaign. Interesting areas with negative aggregates, i.e., locations where the returned amount exceeds that of the contributed, are shown as blue.

Due to its close relation to KDE, we achieve a continuous reconstruction of the distribution of this "value mass" with respect to the two other data attributes $d_{i,a}$ and $d_{i,b}$. This leads to very interesting visualization options for absolute quantities (not just relative densities as with KDE). In Fig. 4, for example, we visualize the distributional characteristics (here with respect to longitude and latitude) of more than 165 000 monetary contributions to the recent Obama campaign; data acknowledged FEC [36]. We achieve a continuous reconstruction of a distribution function that tells in which places how much was contributed. One strange result from this visualization is the identification of locations where the overall aggregation of all c_i values is negative (resulting in blue color), meaning that the average contribution per square mile is a negative amount of dollars. The dataset contains transactions that represent contributions that have not been accepted (and therefore returned, accordingly). One valid explanation for these negative areas is that the agencies have been more meticulously in registering the zip-code for cash returns than the initial contribution (but additional analysis would be required to fully understand this phenomenon).

5 Reconstructing Time

In many cases, and also later in our application context, we are confronted with streaming data from different types of processes. To achieve a truthful visualization of time-dependent data of this type, we need to integrate KDE with a proper representation of the continuous change over time. One approach could be to super-sample the streaming data with respect to time, resulting in a reconstruction based on a large set of kernels. Instead we suggest using a line kernel that amounts to a pre-integrated continuous solution to this problem. Fig. 7

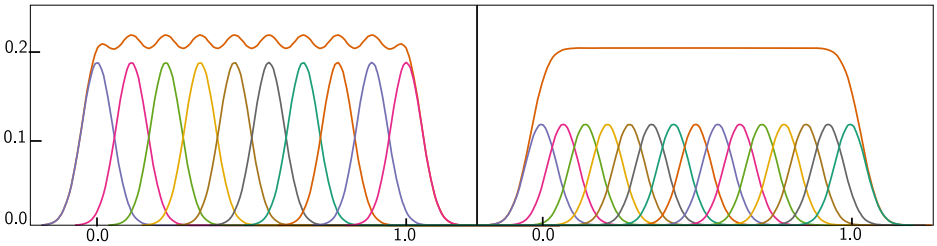


Figure 5: Gaussian kernels with a bandwidth of 0.05 and their combined integral (orange). Left 10 kernels and their sum, and right 15 kernels and their sum. These figures represents the super-sampling approach, whereas Eq. 8 calculates the sum directly.

shows the proxy geometry needed to implement this super-sampling (left) and our line kernel (right).

Accordingly, we adapt kernel density estimation to reflect this reconstruction scheme. We suggest a kernel L_k to reconstruct the contribution of a line (instead of just a point). Then, assuming a dataset of n in-streaming data items, the time reconstruction estimate $\hat{t}(\mathbf{x})$ becomes

$$\hat{t}(\mathbf{x}) = \sum_{k=1}^{n-1} L_k(\mathbf{x})\,. \tag{7}$$

For every two consecutively in-streaming data items $\mathbf{d}_i$ and $\mathbf{d}_{i+1}$, and their associated point locations $\mathbf{p}_i = \mathbf{p}(\mathbf{d}_i)$ and $\mathbf{p}_{i+1} = \mathbf{p}(\mathbf{d}_{i+1})$ in the 2D KDE domain, a line reconstruction kernel L_k is placed that is constructed as follows:

$$L_k(\mathbf{x}) = \int_0^1 c_i K_{\mathbf{H}}\big(\mathbf{x} - ((1-\phi)\mathbf{p}_1 + \phi\mathbf{p}_2)\big)\,\mathrm{d}\phi \tag{8}$$

$K_{\mathbf{H}}$ is one of the kernels that otherwise are used for point reconstruction, in our case we use the normal kernel here. And c_i is a scaling factor for each line segment, i.e., when reconstructing time, the time passed, especially also to support uneven sampling. Eq. 8 is the converged result of distributing point reconstruction kernels evenly along the line segment. The converged result of super-sampling is detailed, as a 1D example, in Fig. 5, whereas Eq. 8 directly evaluates the converged result. Fig. 6 illustrates the distribution of time, when tracing a sequence of four points, or, three edges, each weighted with one second. According to Eq. 8, these three line kernels each contribute a weight of one second, to the total integral, but since the line on top has a shorter distance between vertices, the density here is higher. Further below, in section 7, we present examples from our application case, e.g., in Fig. 9, that was also reconstructed with this approach.

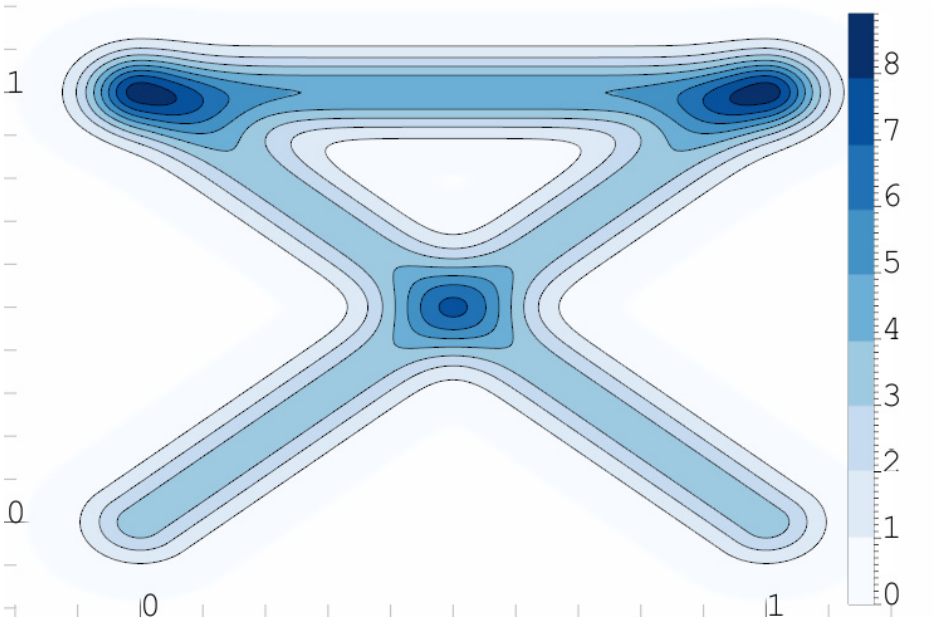


Figure 6: A line kernel density reconstruction of four samples, or three edges. Each edge is weighted by one, e.g., one second, and thus the integral of this entire figure is three. The time density at the top edge is greater than the diagonals, since this distance is smaller, and its weight is the same.

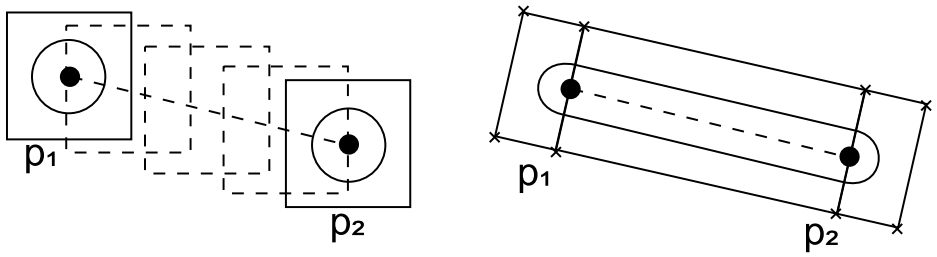


Figure 7: Reconstructing two connected samples in time, on left, super-sampling by filling the space with additional samples, and on right, by drawing a continuous rectangle and two end-caps. Both techniques produce the same result, but our line kernel density estimate does so with a significant efficiency increase.

6 Interactivity and Analysis

Defining interaction with a system requires one to first identify the internal parameters that can be modified, and second, on a higher level, identify the tasks that users would perform on that system. The parameters available in a KDE-based visualization are only data-samples, bandwidth, and viewport. Shneider-

man listed a set of tasks [106] that fit the information visualization workflow, "Overview first, zoom and filter, then details-on-demand". Creating an overview from a KDE is simply ensuring that the shown range of the two dimensions is spanning all samples and choosing an appropriate bandwidth. Zooming and panning are direct manipulations of the viewport, and is closely related to filtering out those samples out of view. Since data investigated often have different units or scale, we suggest that zooming should be allowed individually per axis; however there are cases where an enforced aspect ratio is desired. When the unit of both axes is the same, and the scale is comparable, keeping an aspect ratio of 1:1 would help to not introduce any misleading scale impression. Another case where an enforced aspect ratio is useful is when displaying maps, or lat-lon axes. In this case we enforce an equidistant cylindrical ratio, which is ratio varying on the current viewport's latitude. This ratio ensures that at least the area around the latitude line in the center of the viewport is equal-area [111].

Often, the automatic generation of parameters is more important than interaction, and two examples of automatic parameter generation are (1) generate decent initial / default values, and (2), have parameters generated optimally, creating a nonparametric functionality, and even removing the need for user-interaction. Visualizing large datasets often makes it impossible to create an optimal viewport, showing all the data, which is why zooming and panning is introduced. There are works trying to globally optimize the bandwidth, e.g., the normal scale rule [108], but we find that this factor is highly dependent on the viewport, e.g., if the bandwidth in either dimension is less than a pixel, nothing is shown. Instead of calculating an optimal bandwidth based on the data-sample distribution, we propose a method that is tightly coupled with the viewport, that will update the bandwidth when the viewport changes. In a right-hand system, a viewport is defined by two points, the lower-left $\mathbf{p}_1$ and upper right $\mathbf{p}_2$. The range, $\mathbf{r}$, of this viewport is then $\mathbf{r} = \mathbf{p}_2 - \mathbf{p}_1$. We then define the pixel size, $\mathbf{s}$, as $\mathbf{r}$ divided by the screen size. We then have two observations. One, if the bandwidth $\mathbf{H}$ is less than $\mathbf{s}$, i.e., less than a pixel, the sample is not shown, and thus we recommend $\mathbf{H} > \mathbf{s}$. Second, if the bandwidth is larger, by a factor k, than the range $\mathbf{r}$ of the viewport, the observed result will be a near constant sum of the kernels within, and around, the view. By defining that $k \cdot \mathbf{r} > \mathbf{H} > \mathbf{s}$ we can assert a viewport independent density estimation of the prominent visible features. If we continue to enforce a bandwidth tied to $\mathbf{s}$, i.e., a pixel bound bandwidth, throughout interaction, we can zoom out to aggregate more features for an overview, and zoom in for a more detailed view. An example of this interactively changing bandwidth is shown in Fig. 1, and in the supplementary video. It our experiences, a bandwidth from approx 2 to 20 times that of a pixel, works well, and is in fact representative for all the figures in this paper, relying on line kernels.

The next task is filtering, i.e., showing only a subset of the samples. When dealing with time dependent data, the most common filter allows temporal se-

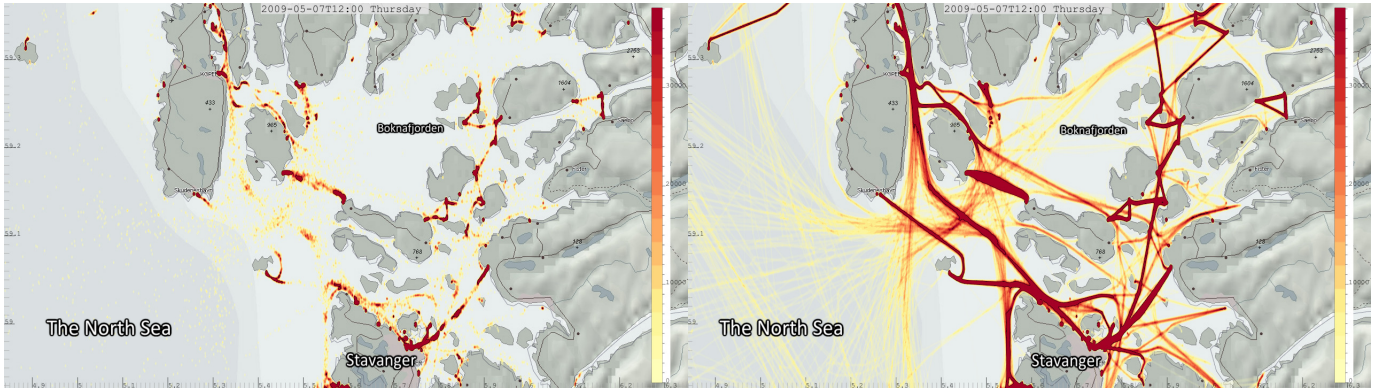


Figure 8: Two KDE-based visualizations using the same bandwidth and data, of ship traffic off the coast in western Norway. The left image shows the position samples as point kernels, and the right image shows the same data using our line reconstruction kernels.

lection and animation. Animating temporal trends can be achieved by setting three attributes, namely, *time*, *time window* and *time step*. Time is the current point in time that samples are shown until. Time window is the how far back in time from *time* samples should be visible, and time step is the increment per animation step. E.g., when showing weekday trends, the time window should be set to 24 hours, but the time step could be set to one hour, so that one would, in a video with 24 fps, get a smooth animation from day to day with a day lasting a second in the animated visualization.

The last task we facilitate is details on demand, however since KDE is not an item based visualization, selection is not available. Instead we propose a simple integration scheme where a bounding box is drawn, and the area within this box is integrated. From Eq. 6 we see that the open integral is the sum of all sample-weights, and similarly the bounded integral gives a sum of the selected region. This interaction enables accurate quantitative analysis of the distribution of this third attribute.

7 Demonstration

In this section we cover three different cases involving streaming data. The first case covers ship traffic off the coast of Norway, the second case investigates data from drilling operations in the petroleum industry, and the third all commercial air traffic in the US spanning two decades from 1987 to 2008.

7.1 AIS Ship Traffic

The Automatic Identification System (AIS) is a radio based system used by ships and other vessels for collision detection and identification. The International Maritime Organization requires all ships with a gross tonnage of 300 or more, in addition to all passenger ships, regardless of size, to be equipped with this system. With the KDE-based visualization approach described here, we enable the real-time filtering, analysis, and rendering of large sets of stored as well as of streaming AIS data. The AIS signals that we study are picked up by the Norwegian shore based network. Here we visualize 14 days of AIS data in which a total of 5000 ships are registered, sending 850 thousand position updates. Willems et al. recently presented a technique for convolving kernels along AIS ship paths [136]. Our visual results are similar to theirs in terms of AIS data. Their implementation, however, takes approx. 10 minutes to compute (data for one day, i.e., 100 000 line segments). Our technique calculates similar results for 14 days (850,000 line segments) in 43 ms (23fps). Because of the rendering speeds we achieve, and since we do not need pre-processing, we can connect to the live feed for streaming AIS data. Fig. 8 shows a small section of the area covered by the Norwegian AIS system, outside the south-western coast. These two figures

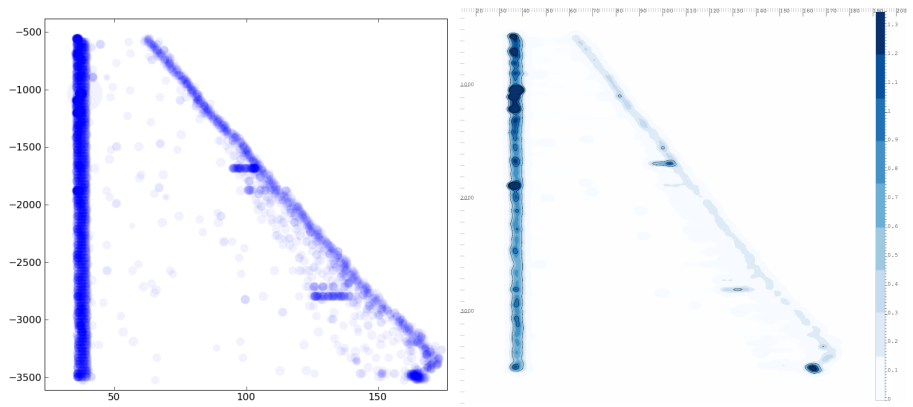


Figure 9: A side by side comparison: an overpopulated scatterplot with semi-transparent points (left) vs. our visualization with line KDE (right). Compared, the bottom of these two gives a clear overview of where time is distributed with regards to hook-load and depth. The dark blue areas to the left indicate non-productive time.

clearly show the advantage of our line kernel reconstruction. On both images, the traffic close to the coast, enclosed by headlands, are clearly defined, but out in the open sea, where the radio signals are weaker, the samples become so sparse that it is hard to detect where the ships move. By zooming to a smaller region, with the sample bandwidth reduced automatically, this sparseness increase even to affect the dense areas in this figure. Using this side by side visualization highlights where the dead zones of the AIS radio system is, and thus where perhaps this could be extended.

Statistics on AIS data have several times proven useful, e.g., when calculating the risk new offshore installations face with respect to collisions. Using our technique we have increased the speed of calculating these probability plots to such a degree that one can interact with them (i.e., recalculate them) at real time speeds (for this dataset, 23 fps). As Norway aims to invest in several new offshore windmill parks, our techniques will enable both manual investigations, and faster and more complex automated placement algorithms.

7.2 Drilling operations

In a project with partners from the Oil and Gas industry we investigate the distribution of time in drilling operations. The dataset that we visualize here contains several measured and derived attributes from this process. In this context we look closer at three of these, namely, *depth*, *hook load*, and *time*. Depth is the length of the drill string that is in the bore hole (and not true vertical depth) and hook load is the measured weight of this drill string. In Fig. 9 we present the visualization of these three attributes in two different versions, a

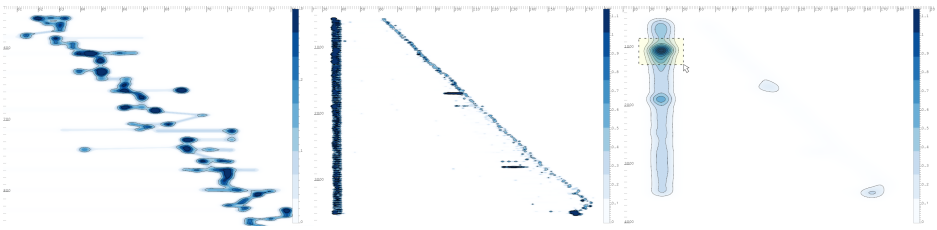


Figure 10: Three line kernel density estimates, showing the distribution of time over depth in a drilling hole (wellbore) and hook-load, the weight of the entire drill string. The leftmost image is a detailed view, with a small kernel, showing the curve with varying tons on the hook, used e.g., to calculate friction. The user then zooms out, and goes to an overview mode with a large kernel, on right, and selects an overwhelming time density, integrates and finds that over an hour was nonproductive at this depth.

regular scatterplot using transparency and a KDE-based visualization using line kernels. The vertical scale is depth, down being deeper, and the horizontal scale is hook load. The most prominent visible features are the two bands, one vertical and one diagonal. The vertical band, at approx. 35 tons, is the weight of the hook when the drill string is not attached to it, and is thus an indicator of the time spent attaching or detaching a new pipe segment to/from the string. The diagonal band is the weight when the drill string is attached to the hook, indicating weight increasing with depth, since there are more pipes attached to the hook. This dataset was acquired when the drilling crew decided to pull the entire string up, from 3500 meters down. This operation is performed every time there is something wrong, or, they want to set a new casing, or change the drill bit. It is important to do this as fast as possible, as time efficiency is paramount to have a good return on investment. When presented to the domain engineers, the first feature discussed was the visualization of unscheduled stops, shown as local peaks. To analyze further, the biggest of these, at about 1000 meters, was zoomed onto (see Fig. 10) and the integral shows a total of one hour, as compared to normally approx. two minutes for removing a 90 feet pipe. One scenario that makes good use of this tool is for the onshore team that monitors the ongoing process, or for the change of shifts, where a new team takes over the drilling, and they would need to get an overview of the recent history of progress and events.

7.3 Commercial Air Traffic

In this section we show how our line kernel density estimate enables insights into a dataset containing all commercial air traffic in US, from October 1987 to April 2008. This dataset [7] contains 120 million flights and makes out 12 gigabytes. The distances flown are calculated by Haversine distance from airport to airport, and goes from 16 trips to the sun and back in 1987 to 28 round-trips in 2007. One interesting note about the summary of all flights is that while the total flight

hours shows an increase of 172% from 1988 to 2007, the number of takeoffs only increased by 142% in the same period, i.e., the more recent average flights travels longer.

This dataset is particularly interesting to investigate using line kernel density estimation (as opposed to regular KDE) because of both the large spatial distance between points. As defined here, one flight is a scheduled takeoff; this dataset contains the origin and destination airport of all flights. From the airport codes and all actual takeoff and landing times we created a new dataset. This dataset is a temporal line-segment dataset. A temporal line-segment consists of two points with values for latitude, longitude and time, each.

Our prototype can show temporal animations at real time, concurrently with interaction, which both require reconstruction of the KDE for every frame. An example interaction is shown in Fig. 1, where the kernel size/bandwidth of the estimate is tied to pixel size, instead of, e.g., km. This bandwidth enables the user to zoom in, while simultaneously refining spatial information. This Fig. 1, contains the automatic aggregated flight hours over the Bay Area at the initial zoom level, and after zooming in, can determine the distribution among the different airports, and their respective distributions along the different cardinal directions as such.

The top row of Fig. 11 shows hour by hour as dusk moves over the US, the air traffic picks up from east to west, a pattern that repeats itself at night, as well. The bottom row of Fig. 11 shows a more dramatic pattern, at September 11th, 2001.

8 Technical Details and Accuracy

In the following, we discuss how we implemented the above presented approach on graphics hardware and discuss performance and accuracy of this solution.

8.1 Kernel Density Estimation on the GPU

The use of modern GPU-accelerated techniques in data visualization is a promising step [37], especially since interactive visual analysis relies on interaction, and thus on interactive rendering. In our prototype we developed a two step technique for computing and visualizing KDE. The first step is to generate a floating point field by evaluating the 2D KDE equation, and the second step is to appropriately visualize this KDE field with one of several options.

Calculating KDE on the GPU requires the support of floating point, or double precision textures, as we need to store results with an appropriate precision. Evaluating the 2D KDE function to a 2D matrix (a texture), can be done in one of two ways, with one cell being one element/texel in our matrix with the properties of a value `v` and a position `p`:

Figure 11: Temporal animation of air traffic on the 10th and 11th of September 2001. The top row shows a normal pattern of how the traffic evolves, following the timezones. These views show a time-window of the two hours leading up to the given time. The bottom row all traffic is cut short, lasting several days, due to the tragic events at this date.

```
a)  for c in cells:
      for k in kernels:
        c.v+=k.eval(c.p)

b)  for k in kernels:
      for c in cells:
        c.v+=k.eval(c.p)
```

I.e., we can either first iterate over the grid cells, or over the kernels, respectively. The latter case is identical to rasterizing on the GPU, and thus this is our selected approach. To create the result, we first allocate a grid, as a 2D *frame buffer object* (FBO), with floating point precision. Then, with this FBO bound, we render all the kernels. All of them are then aggregated with an additive blend operator. To create an optimized implementation, we allow for an approximation of KDE by limiting the extent of all kernels (we will return to this subject in the next section). To further optimize this implementation as well as, to enable distinctive kernels, we pre-compute the kernel and store them as a floating point texture. The geometry needed for point kernels can be created by either using the point sprite extension, drawing quads, or more efficiently using geometry shaders. The use of point sprites or geometry shaders reduces the necessary vertices to one. Fig. 7 shows the necessary vertices needed to construct a line kernel, which we construct out of three quads. Here the use of a geometry shader reduces the necessary vertices to two, p_1 and p_2.

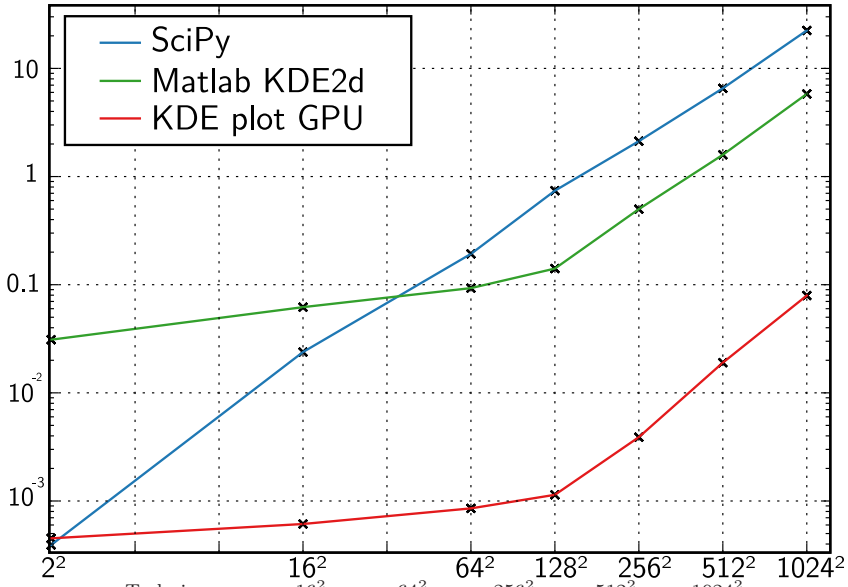
To enable a fair comparison to other KDE algorithms, we have created a Python interface, that stores the result as NumPy arrays. Fig. 12 shows the result of a comparison of three different algorithms for the 2D kernel density estimation in the Iris dataset, containing 150 samples. The three different implementations we used are the SciPy [67] implementation, a Matlab™ file implemented by Botev [15], and our implementation on the GPU. As this table shows, there is a significant, up to approx 300 times large speed-up, e.g., compared to the Matlab implementation for the 1024^2 grid.

8.2 Error Estimation

In this section we investigate the computational accuracy of our GPU-based KDE (based on a Gaussian kernel as discussed in Sec. 3), in addition to an overall discussion on the errors or drawbacks that can arise using KDE. As a kernel with infinite extent, the Gaussian is defined over the entire real line $\mathbb{R}$. As an approximation, a windowed kernel can be considered, e.g., by truncation [26]. To investigate how good bounded approximations are, we look at their integral for comparison. The finite integral of the Gaussian 2D product kernel, $N(x, y) = \frac{1}{2\pi} e^{-\left((x^2+y^2)/2\right)}$, is:

$$\int_{-n}^{n}\int_{-n}^{n} N(x,y)dydx = \operatorname{erf}\left(\frac{n}{\sqrt{2}}\right)^2 \tag{9}$$

where erf is the "error function" (encountered when integrating the normal distribution). Using Eq. 9, we can calculate that the use of a texture with interval



Technique	16^2	64^2	256^2	512^2	1024^2
KDE-plot GPU	6.1 E−4	8.5 E−4	3.9 E−3	1.9 E−2	7.9 E−2
matlab KDE2D	6.2 E−2	9.3 E−2	0.5	1.6	5.8
SciPy	2.4 E−2	0.19	2.1	6.5	22.4

Figure 12: Run times (in seconds) for evaluating grids of different sizes, for three different implementations of kernel density estimation, all using same dataset, kernel and bandwidth. *KDE-plot GPU* is our proposed technique.

n/interval	1	2	3	4	5
error	0.53	8.9 E−2	5.39 E−3	1.27 E−4	1.15 E−6

Table 1: Error introduced by using a truncated Gaussian.

Technique	2^2	4^2	8^2	64^2	128^2
Central	0.97	0.163	1.33 E−5	1.12 E−6	1.14 E−6
Preintegrated	0.0	0.0	0.0	0.0	0.0

Table 2: Error introduced by integrating (summing) textures of different sizes. Results show one minus integral. ▮

$[-n, n]^2$ will result in an error as shown in table 1. In cases were normalized kernels are used, the interval $[-5, 5]^2$ with an error of $1.15E - 6$ is sufficient. However since we are scaling every kernel by a factor, this error would also be scaled linearly.

When representing a kernel K as a discretized texture, the integral is the sum of all texels, multiplied by the texels' size (e.g., on an interval $[-5, 5]^2$ and on an 128^2 texture: $10^2/(128^2)$). Using a discretized 2D Gaussian in the interval $[-5, 5]^2$ can ideally never achieve a better integral than eq. 9, but we now look into

the actual integrals using different techniques. We compare two techniques for creating and integrating kernel textures. The first, called *central*, gives every texel its value after evaluating K with its central position. In the second technique, called *preintegrated*, the integral over the the area spanned by the texel is assigned to the texel. Table 2 shows the errors introduced using different techniques and texture sizes. The errors presented for the central technique will, for larger texture sizes, converge towards the error presented in table 1.

Kernel Density Estimates, reconstruct a continuous distribution from a discrete set of samples, essentially by smoothing. In several cases, this smoothing can introduce errors. As an example of this smoothing error, we can think of a shipping lane, where the vessels are passing through a very narrow straight. If we smooth out these vessel paths, we have a low tolerance, before we introduce a probability of finding vessels on land. While not covered in this paper, there is several existing works, on variable kernel density estimation, on how to specify an individual, and optimal bandwidth, for every sample. In our implementation of the line kernel, defined in Eq. 8, we allow for an individual bandwidth per sample, enabling support for varable kernel density estimation. However, for purposes on streaming data, without pre-processing, this individual bandwidth cannot be implmented, in a trivial fashion.

Another source of errors lies in our restriction to a simple bandwidth matrix, in Eq.4. If the data modeled contains a diagonal distribution, the correct kernel to use would be one with skew, and thus cannot be modeled using our simplified bandwidth. It is however trivial to extend, the line kernel to allow the full bandwidth matrix. Our rationale for not utilizing this however, lies in the lack of preprocessing, so, we, because of streaming data, cannot pre-process to find this optimal bandwith matrix.

9 Summary and Conclusions

In this paper, we discuss the challenge of intuitively visualizing large amounts of discrete data samples. We discuss a KDE-based visualization, defined from the statistical concept of kernel density estimation (KDE), as an elegant solution. We adapt this concept to also allow for investigating the distributional characteristics of an additional, third attribute over two dimensions. Additionally, we show how KDE-based visualizations can be extended to visualize the distribution of time in the context of streaming data (with a new type of a line kernel). We explain and demonstrate how KDE-based visualizations can be computed on the GPU, leading to speed-up factors around 100 (and up to approx 300 in one of our cases). We briefly report on our prototype in the maritime, the oil & gas domain, and air traffic and show that useful results are achieved.

We demonstrate that due to our improvements to both regular and streaming KDE-based visualizations, utilizing modern GPUs, it is now possible to utilize

advanced concepts from statistics for improved visual data exploration and analysis, for large data at interactive speeds. With respect to KDE, in particular, it would be great to see more interesting related future work in visualization.

10 acknowledgements

The work presented here is a part of the project "e-Centre Laboratory for Automated Drilling Processes" (eLAD), participated by International Research Institute of Stavanger, Christian Michelsen Research and Institute for Energy Technology. The eLAD project is funded by grants from the Research Council of Norway (Petromaks Project 176018/S30, 2007-2011), StatoilHydro ASA and ConocoPhillips Norway. Furthermore we acknowledge the Norwegian Coastal Administration for supplying access to the AIS.

Paper B

Curve Density Estimates

Ove Daae Lampe[1,2], and Helwig Hauser[1]

[1]Department of Informatics, University of Bergen, Norway
[2]Christian Michelsen Research, Norway

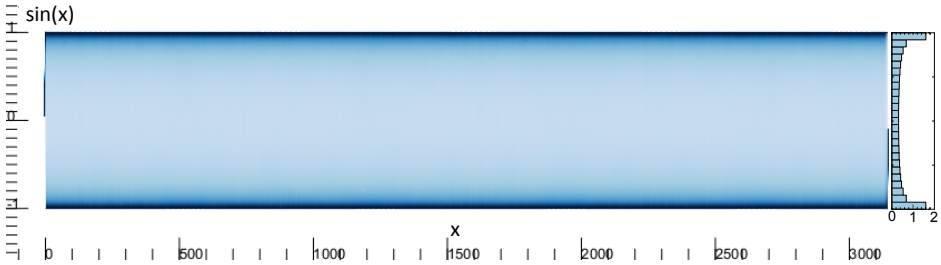


Figure 1: The Curve Density Estimate of a high frequency sine curve with a normalized histogram of evaluated values densely sampled along the x axis. Our continuous representation of this curve closely matches that of the histogram.

Abstract

In this work, we present a technique based on kernel density estimation for rendering smooth curves. With this approach, we produce uncluttered and expressive pictures, revealing frequency information about one, or, multiple curves, independent of the level of detail in the data, the zoom level, and the screen resolution. With this technique the visual representation scales seamlessly from an exact line drawing, (for low-frequency/low-complexity curves) to a probability density estimate for more intricate situations. This scale-independence facilitates displays based on non-linear time, enabling high-resolution

This article was published in *Proceedings of Eurographics/IEEE-VGTC Symp. on Visualization (EuroVis 2011)*, 30(3), pages 633–642, 2011, and presented at EuroVis in Bergen, Norway by Ove Daae Lampe.

accuracy of recent values, accompanied by long historical series for context. We demonstrate the functionality of this approach in the context of prediction scenarios and in the context of streaming data.

1 Introduction

In the context of time-dependent data the drawing of function graphs is one of the most natural and at the same time one of the most effective data visualization techniques. As long as the spatial complexity of the graph is limited, this immediate translation of data into a graph is straight forward and provides intuitive results. If the curve to draw, however, becomes very long or the spatial complexity increases, for example when considering a fractal curve, then the simple plotting of such a curve or graph will likely result in problems with overdraw and cluttering. This overdraw in an example graph led us to the question: Why did our regular graph of a sine curve look so different when its samples where drawn in a scatterplot instead? (see Figure 2 a and c). The scatterplot, when drawn with transparency, resembles the histogram of these values, as shown in Figure 3. This histogram shows the distribution of the evaluated values, but the curve representation completely obscures this distribution, even when applying transparency, as shown in Figure 2b. In this paper, we investigate an alternative way of rendering such curves, that does not display the same problems, but keep the clear benefits of the regular curve.

As, perhaps, a very trivial summary, a function graph displays a single measured value, on one axis, with its continuous changes over another axis. Unless the changes are piecewise continuous, a curve is not an appropriate choice of visualization.

One of the biggest challenges when drawing function graphs is that they are mainly useful for displaying frequencies that, on the extreme, is at least greater than the pixel with of the display. The common way to deal with this is to either constrain/zoom in on the axis, or to aggregate values, to "smooth" out rapid changes. In Figure 4, aggregated stock prices for Intel are shown, with the black curve being the center-shifted moving mean, enveloped in the curves' standard deviation. The outer polygon is the moving max-min. This enveloped curve, as described by Miksch et al. [84], captures the overall movement of the underlying data very well, and its standard deviation polygon reveals important information about the frequency or stability of the smoothed data. This method works best when the data is close to the normal distribution (or at least unimodal).In the case of a bi- or multimodal data distribution, however, this aggregation looses certain expressiveness. The visualized mean can easily associate with highly improbably data values, for example, in the middle between two modes. Moreover, the moving mean relies on a certain window, which either provides lagging results or it is undefined for the latest values.

As a simple example we consider a sine curve from zero to a number larger than the amount of available pixels in the horizontal direction. A naïve approach to display this curve is shown in Figure 2a, which suffers from overdraw, and would mainly only display the extent of the curve. A first approach on how to solve this problem could be to apply transparency, shown in Figure 2b. The transparency

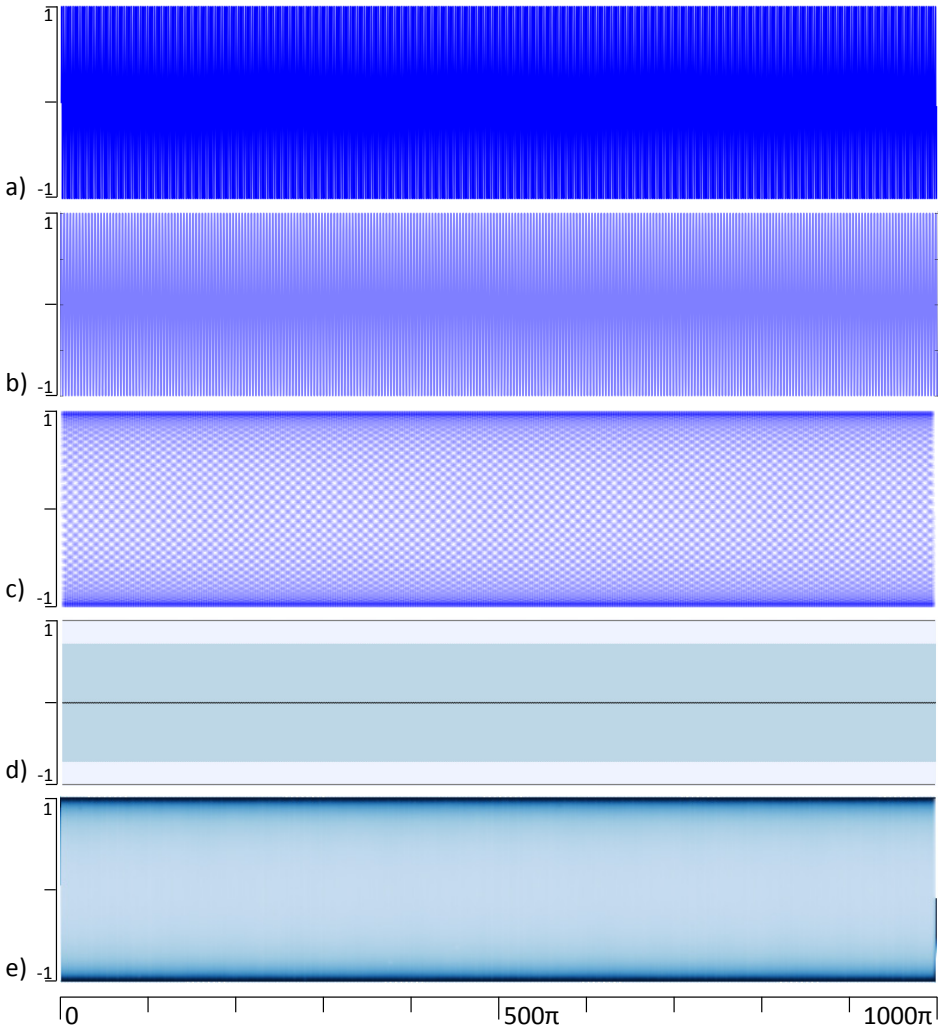


Figure 2: Figures displaying the sine curve from zero to 1000π. In the top figure, a, an opaque line is used, and because of overdraw, displays only the extent of the function. In the second figure, b, a transparent line is used. The third figure, c, is a scatter-plot of the samples drawn transparent, and shows the same distribution as the histogram. The fourth figure, d, is aggregated with moving mean, standard deviation and extent. As opposed to Figure 4, this data is unsuitable for this type of aggregation. In the bottom figure, our technique, the Curve Density Estimate, is applied, and the distribution corresponds with that found in the histogram in Figure 3.

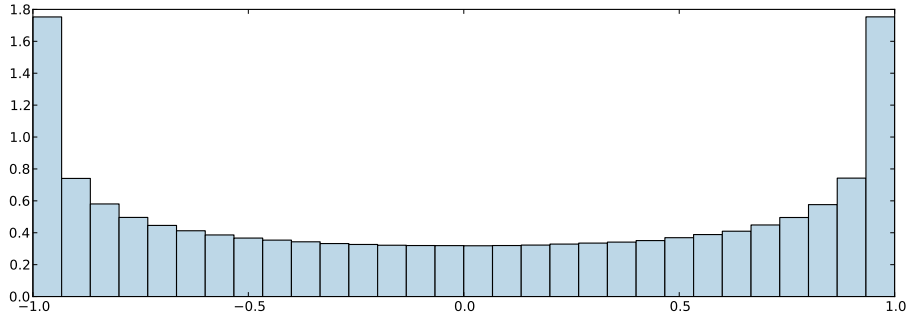


Figure 3: 30 bins histogram of $y = \sin(x)$ for regularly sampled values of x.

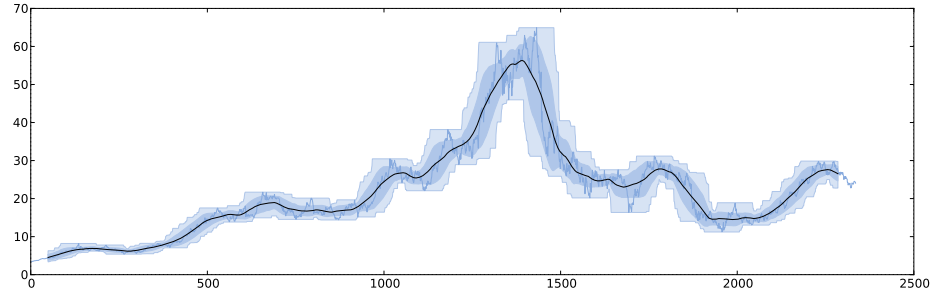


Figure 4: Intel opening stock price with a moving average, standard deviation and extent.

could correctly display the amount of overdraw, but this does not correctly display the distribution of the curve. The distribution of a curve, alternatively described as its continuous histogram by Bachthaler and Weiskopf [9], is found by taking regular samples along its parameter axis, and inserting evaluated values in a histogram. The histogram for the sine curve is shown in Figure 3. It is worth noting that the curve with transparency will have a single visible mode at zero, which is almost the opposite of what the histogram indicates, with two modes at one and minus one.

A second technique on how to deal with this high frequency sine curve could be to first aggregate it. In Figure 2d, we show, similar to Figure 4, the moving average, standard deviation, and extent. Using a sufficiently large window, the average of the sine curve is stable at zero, and its standard deviation is constant. Part of the problem is that this way a model, with a normal distribution, is enforced onto the data. If there is a mismatch between the assumed model and the data, such a visualization will not be expressive.

With our technique applied to the sine curve, as shown in Figure 1 and Figure 2e, we recreate the distribution without any prior knowledge of model, and

will do so independent of frequency, zoom level and screen resolution. That said, we do not propose to replace the aggregation techniques, as in Figure 4 or other techniques, where the model is known, but provide a default view, that can either be used before the model is established (if it exist), or to investigate how well a selected model fits the data.

With this paper we introduce a novel way of displaying function graphs, that also supports:

- Graphs with a frequency higher than the pixel-width of its display
- Smooth transition between high frequency areas and single line curve
- The creation of a probability density estimate that does not assume a normal distribution of values
- The probability density of both single and multiple curves (and a mix between those)

In the following, we first discuss related work, then move on to the theoretical details of curve density estimates, before we add technical implementation details. Lastly, we apply our technique to real world data before providing the summary and conclusions.

2 Related Work

Existing techniques, improving or extending the curve, fall briefly into the categories: compact views, overdraw views and, distribution views.

Compact Views: By utilizing techniques to compress the value axis, the aspect ratio is improved such that longer time-series can be shown on less space without contracting the time axis, and thus avoid the frequency problem. Saito et al. [102] designed a compact graph view, that utilizes a colored banding to overlay multiple ranges of the curve on top of each other. Using this banding, which was further refined into the horizon graph by Panopticon [98], a precise value can be read out, while reducing the physical height down to an eighth. When the value range of a process is known, and also can be defined in levels, such as low, normal and high, all values in these ranges can be replaced with colors to produce a compact visualization, as described by Bade et al. [10]. They provide an interesting example of body temperature graphs, where there are clearly defined normal levels. Another compact graph visualization are the Sparklines as introduced by Tufte [120], which strips the curve down to a text-line sized graph, that can even be included mid-text. As a separate thread of compact visualization techniques, is the pixel based category, where each sample is displayed using a colored pixel. In 2008, Hao et al. [52] provided an evaluation on how best to place such pixels, while keeping temporal coherence. While not quite a compact view, Kincaid proposed combining high frequency time series with a

focus+context interaction [74]. This interction provides both an overview, and a detailed view down to the individual samples.

Overdraw Views: Techniques that deal with visual clutter of high frequency by introducing schemes to blend multiple overdraws. Most visualization packages allows the user to specify different opacities, effectively implementing an overdraw view. In 2002, Jerding and Stasko introduced the Information Mural [64] to deal with high frequency graphs and other cluttered 2D visualizations. This technique downscales large, original, and uncluttered views to miniatures, while counting the overdraws to each pixel. This overdraw count is then used to apply a greyscale color.

Distribution Views: Techniques that by aggregation deduce the distribution of a single, or multiple curves. Hochheiser and Shneiderman [57] utilized envelopes that displayed the full extent of curves, in their TimeSearcher application. In 2004, Kosara et al. [77] described the TimeHistogram, where the time axis was divided into intervals, and a separate histogram was calculated for each of these intervals. These histograms, with colored 1D representations, was then in turn displayed along the time axis. In the work by Muigg et al. [87] multiple curves where binned while aggregating both the count and the directions, to create a visualization close to that of a flow field, utilizing line integral convolution (LIC) to overlay direction on top of the frequency. Bade et al. [10] introduced an extension to the information mural [64], that adds the median, the 25 and 75 percentiles and extent. Which is similar to BinX [14] which visualizes long time series by binning along the time axis at different levels of aggregation and then displays mean, minimum, maximum value, and standard deviation per bin. Johansson et al. [66] discussed a blending scheme to introduce temporal changes in parallel coordinate plots (PCP). Their implementation aggregates continuous changes on top of each other forming, what can be described as a continuous 1D histogram from discrete samples. This 1D histogram is then the basis for creating a polygon that is drawn to the next axis in the PCP. Feng et al. [38] also introduced an extension of PCP, that is the result of mapping the 2D KDE between each axis, into its corresponding parallel coordinate version. Furthermore, Feng et al. [38] also introduced several enhancements to interaction and brushing techniques to better suite frequency data.

The technique proposed in this paper is an extension of our previous work [28], where we first introduced the concept of a line kernel to kernel density estimation. The line kernel is used to reconstruct continuous changes, by connecting consecutive samples, forming an elongated kernel, and that integrates up to one independent of the distance between samples. In this work we extend this line kernel, and show how to reduce it into an exact and continuous, parametric formulation. In our previous work, we utilized a table of pre-integrated convolved results, whereas this extension allow us to directly evaluate the exact result. Additionally, in this work, we introduce a curve visualization, called *curve density estimates* (CDE), that provide distributional characteristics along the time axis,

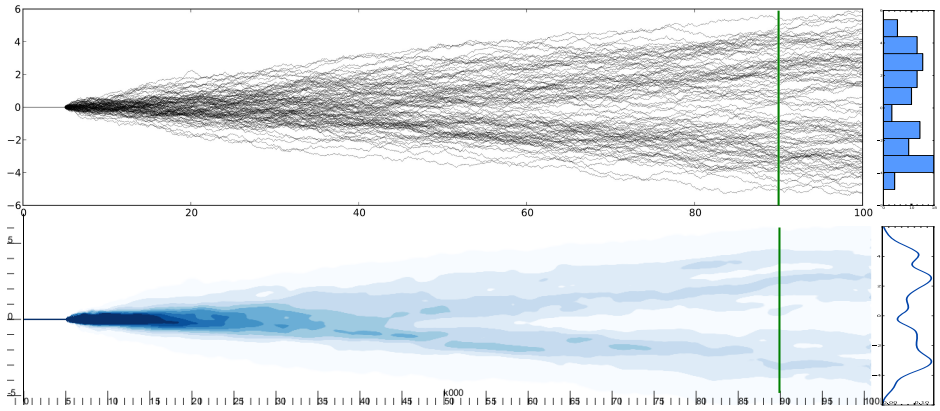


Figure 5: 100 cumulative random curves with a slight bimodal trend. The top graph show the curves with slight transparency. All the samples at the green line, at $x = 90$, are drawn using a histogram and a 1D KDE in to the right. The graph to the bottom shows the CDE. Note how the 1D KDE corresponds to the green line drawn over the CDE as well.

comparable to the concept of a continuous 1D KDE. This visualization is enabled by a moving column based normalization scheme further detailed in Section. 3. Several other extensions are also provided here, over our previous work, e.g., non-linear time, single curve to multiple curves transition.

3 Curve Density Estimates (CDE)

The main rationale behind the use of kernel density estimation (KDE) as a basis is that it does not impose any model on the data. Given a discrete set of samples, with an appropriate bandwidth and kernel, KDE can truthfully approximate any probability density estimate (PDE). For an extensive overview of KDE we refer to Silverman [108]. The KDE is defined as the sum of a number of kernels, one kernel per sample. With $(x_1, x_2, ..., x_n)$ being samples corresponding to an unknown density f, the according KDE is defined as

$$\hat{f}_h(x) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i) \quad = \frac{1}{nh}\sum_{i=1}^{n} K\Big(\frac{x - x_i}{h}\Big), \tag{1}$$

with K a suitable kernel. As the kernel, often the normal distribution, $N(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\ e^{-\frac{(x-\mu)^2}{2\sigma^2}}$, is used, with μ being the mean, σ^2 the variance, and h the bandwidth, or kernel size.

The blue vertical graph to the bottom right of Figure 5 shows an 1D KDE. This KDE is created from the set of points where the black curves intersect the

green line in the upper-left graph in the same figure. This 1D KDE clearly reveals the bimodal nature of this dataset. Our curve density estimates (CDE) in the lower graph in Figure 5, can be interpreted as a continuous series of these 1D KDEs. However, instead of modeling our solution by expanding 1D KDEs, we find a solution via a 2D KDE. A standard 2D KDE can be created using the unconnected sample-points from the dataset, inserted into Eq. 1. This approach will not create a continuous distribution when the samples get far apart, but rather be the distribution of the previously mentioned scatterplot (2c). The consecutive samples in the time series represent a continuous change from one value to the next, and thus the probability, given two samples, should not be 0.5 at each sample, but rather be distributed evenly from one to the next. We achieve this by building upon a line kernel L_k defined by two consecutive data samples, and their positions $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ [28]:

$$L_k(\mathbf{x}) = \int_0^1 c_i K_{\mathbf{H}}\big(\mathbf{x} - ((1-\phi)\mathbf{p}_i + \phi\mathbf{p}_{i+1})\big)\,\mathrm{d}\phi, \tag{2}$$

with $K_{\mathbf{H}}$ being the 2D normal distribution kernel. To enable a proper reconstruction from uneven sampling in time, we insert the elapsed time between the two samples in the scaling factor c_i.

We now can reduce Eq. 2 to 1D by only considering values on the line defined by $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$. We name this 1D equation $L_{k1D}(x)$. Furthermore we define the 2D points $\mathbf{p}_i$ and $\mathbf{p}_{i+1}$ to their 1D equivalents (they are per definition *on* this line), q_i and q_{i+1}, respectively. This 1D line kernel is then defined as the integral of Gaussians placed along a line segment. So for any point x, we observe that $L_{k1D}(x)$ is defined by the sum of these kernels, and that all those kernels incrementally have a mean/μ that is greater and greater than x. By turning this problem around, we deduce that the integral on one position of kernels with its mean moving away, is equal to the finite integral over a single kernel. The integral of the normal distribution is a cumulative distribution function (cdf). This distribution function is defined by

$$\mathrm{cdf}(\mathrm{x}, \mu, \sigma) = \frac{1}{2}\left(1 + \mathrm{erf}\big(\frac{x-\mu}{\sqrt{2\sigma^2}}\big)\right). \tag{3}$$

Considering a point x where $x < q_1$, and for explanation purposes a $q_2 \to \infty$. At this point x, $f(x) = \mathrm{cdf}(\mathrm{x}, \mathrm{q}_1, \sigma)$, since it is equal to the unbound integral of all the kernels starting from q_1 towards ∞. However, since q_2 is actually a finite value and we do not have any contribution from kernels beyond this point, we have to remove this from our equation. The contribution from all kernels starting at q_2 going towards ∞ is similarly, $f(x) = \mathrm{cdf}(\mathrm{x}, \mathrm{q}_2, \sigma)$. We then conclude, that for the two points q_1 and q_2, where $q_1 < q_2$, the line kernel, in 1D, is given by:

$$L_{k1D}(x) = \frac{1}{|q_2 - q_1|}\big(\mathrm{cdf}(\mathrm{x}, \mathrm{q}_1, \sigma) - \mathrm{cdf}(\mathrm{x}, \mathrm{q}_2, \sigma)\big), \tag{4}$$

with $|q_2 - q_1|$, the length between these points, applied for normalization, since

$$\int \mathrm{cdf}(\mathrm{x}, \mathrm{q}_1, \sigma) - \mathrm{cdf}(\mathrm{x}, \mathrm{q}_2, \sigma) dx = q_2 - q_1. \tag{5}$$

One important quality of this line kernel is when q_1 approaches q_2

$$\lim_{q_1 \to q_2} L_{k1D}(x) = N(x) \tag{6}$$

the line kernel approaches the normal distribution, $N(x)$, an observation which was also previously made by Kniss et al. [76].

The next step, is to expand this 1D line kernel, over to our 2D case again. In our previous work [28], we relied on the product kernel to define the line kernel. Here, to expand our 1D parametric line kernel to 2D, we also rely on a product kernel, but we let the first kernel be our 1D line kernel, and the other, the normal distribution. Let $\mathbf{w}$ be the point $\mathbf{x}$ projected onto the line L defined by $\mathbf{p}_1$ and $\mathbf{p}_2$, i.e., $\mathbf{w} = \mathbf{x} + |\mathbf{r} \cdot (\mathbf{x} - \mathbf{p}_1)|\mathbf{r}$, with $\mathbf{r}$ a unit vector perpendicular to the line. Then let u be the distance $|\mathbf{p}_1 - \mathbf{w}|$ and v the distance $|\mathbf{x} - \mathbf{w}|$. This then gives the new and parametric 2D definition of the line kernel:

$$L_k(\mathbf{x}) = c_i L_{k1D}(u) \cdot N(v) \tag{7}$$

The KDE using our line kernel that will continuously reconstruct the sample points $(\mathbf{p}_1, \mathbf{p}_2, ..., \mathbf{p}_n)$ is defined by

$$f_{Lk}(\mathbf{x}) = \sum_{i=0}^{n-1} L_k(\mathbf{x}, \mathbf{p}_i, \mathbf{p}_{i+1}) \tag{8}$$

By evaluating this KDE we get a density field with the integral

$$\int f_{Lk}(\mathbf{x}) d\mathbf{x} = \sum_{i=1}^{n} c_i, \tag{9}$$

since individually, all kernels integrate up to one, but are scaled by c_i. Usually, to create a probability density estimate, we would normalize by this integral, but instead we propose to normalize it after rasterization, and then only column by column, individually. Given the 2D grid, G, evaluated by $f_{Lk}(\mathbf{x})$, we create a column-normalized grid G_n by,

$$G_n[i, j] = G[i, j] / \sum_{\hat{j}=0}^{h} G[i, \hat{j}], \tag{10}$$

where h denotes the height of the grid. Figure 6a, displays two curves, that coincide before separating, and Figure 6b the rasterized result, after this column-wise

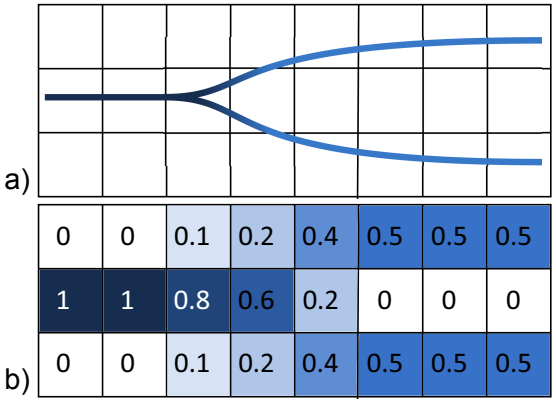


Figure 6: Two coinciding curves splitting up, in a, and the rasterized result after normalization, in b. Note that all columns sum up to one.

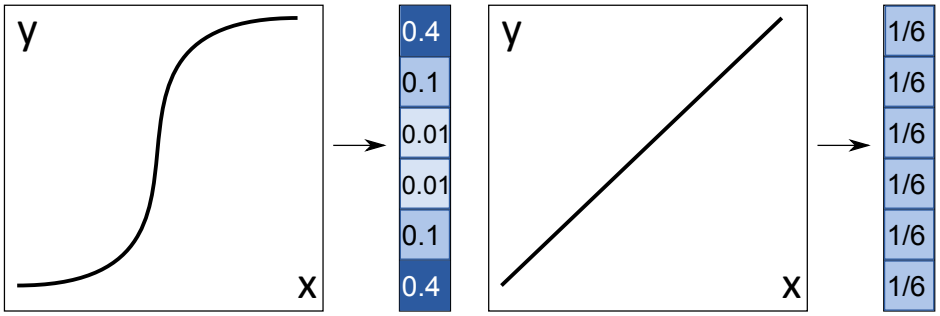


Figure 7: Two different curves, on the left, and their corresponding views, on right, after being rescaled down to one column.

normalization. The rationale for this normalization is to, first, have an intuitive number indicating how much time the curves spent where, and secondly be able to interpret every column as a 1D probability density estimate. A one indicating that all the curves was here 100% of the time, and 0.5, 50%, regardless of how many curves are used, and importantly, if non-linear time is used, regardless of how compressed time is. Utilizing this normalization will render single curves, with small changes, with the most intense values from the chosen color map, and also effectively applying anti-aliasing. However, when the curve reside in the same column, with large changes, e.g., when zooming out, or showing long time-series, the normalization will give the probability density/continuous histogram of where the curve "spent its time". Figure 7 shows two different curves, and their rasterized results after rendering them into a single column.

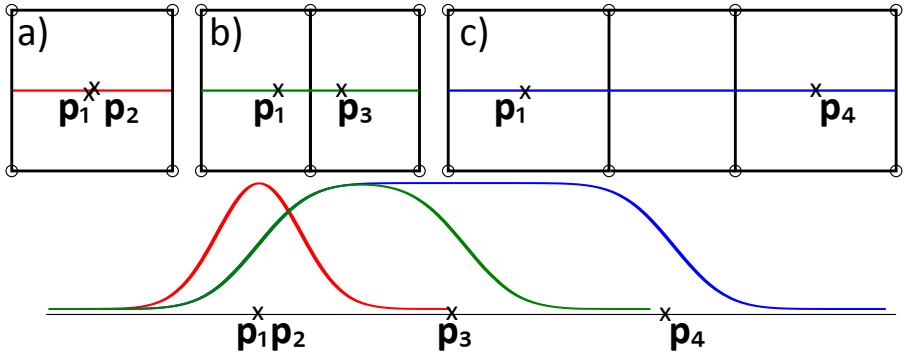


Figure 8: The three different proxy geometry schemes for reconstructing line kernels. The circles indicate the vertices needed. The colored curves below, depicts the evaluated kernel in the corresponding cross-sections above (not normalized).

4 Technical Details

While we see the usefulness of this technique in off-line renderings for static displays, we will, in this section, first address the challenges we face when either rendering real-time streaming data, or need interactivity, e.g. zooming and panning. In order to overcome the challenges and achieve good frame-rates, we offload all the calculation to the GPU. When rendering large time-series, we store vertex arrays, containing the samples, in GPU residing memory, in chunks sorted temporally. Because of this temporal ordering, we perform an intersection test, with the chunks' span vs. that of the view, to determine if it should be rendered or not. The chunk containing the most recent values, however, are stored in main memory, to be able to constantly append new values to it. To minimize the memory usage, the memory structure only contains the samples once, and the proxy geometry, needed by the line kernels, is constructed in a geometry shader step. The geometry shaders input are the consecutive samples, using the GL_LINE_STRIP_ADJACENCY, and it will output the proxy geometry in three distinct ways.

In the first case, a single, unconnected kernel is constructed as a quad. This case is used when the two consecutive samples' distance is less than a threshold $|\mathbf{p}_i - \mathbf{p}_{i+1}| \leq \epsilon$, and that threshold, expressed in terms of pixels, should be one. In our experience, however, there is no significant visual change when increasing ϵ to three pixels (given that the bandwidth is larger than that as well). This case is depicted in Figure 8a.

In the second case, when $\epsilon < |\mathbf{p}_i - \mathbf{p}_{i+1}| \leq c\sigma^2$, we have a line kernel, but one that have a single mode, or maxima. This occurs for $c \approx 5$ (see our previous

work[28] for a discussion on this clamping, and for the errors introduced), when distance is approx. five times that of the bandwidth, since the line kernel L_k is in effect the sum of multiple normal distributions, and its edge will follow the *cdf* function. This line kernel consists of two quads, meeting at the center-point between the points $(\mathbf{p}_i + \mathbf{p}_{i+1})/2$. This case is depicted in Figure 8b.

The third case, is when the distance between the points is sufficiently large, i.e., $|\mathbf{p}_i - \mathbf{p}_{i+1}| > c\sigma^2$, to have a "flat" region between them. In this case the proxy geometry consists of three quads, two for the end caps, and one for the continuous region in between. This case is depicted in Figure 8c. In this case, we can simplify the calculation of the line kernel L_k, to only include the normal kernel perpendicular to the centerline, for the middle segment.

Another usage for the geometry shader, in addition to the three previous cases, is applied when non-linear axes are used. The geometry shader evaluates, according to the distance between the two points, the error introduced by a single linear kernel, and performs a subdivision if needed.

Now, after the geometry is constructed, the kernels evaluate Eq. 7, and their sum is stored in a 32bit floating point texture. The fragment shader, parameterized with u and v, respectively, along and across the geometry, evaluating Eq. 7, calculates $N(u)$ using a table lookup, and the $L_{k1D}(v)$ using existing erf hardware accelerated GLSL implementation. Then, two steps remain, namely, the column normalization, and the application of a color-map. Before the normalization of the columns, however, we first need to calculate the sum per column. We store these sums in a 1D floating point texture, with the same width as the source image, the 2D rasterized grid. To calculate the sum, we first bind the 1D texture as a frame buffered object (FBO), as the render target, and then in a fragment shader, iterate over all texels in the corresponding column of the 2D texture, calculating the sum. As the last step, we apply the normalization division per fragment, while simultaneously applying the color-map.

5 Applications

In this section we cover a varied set of applications for the technique of curve density estimates (CDE). With these applications we show both the generality of the proposed technique while also highlighting specific areas where this technique can outperform the current state of art. Some of these examples are also covered as videos in the supplemental material to demonstrate the interactivity.

5.1 High Frequency Curves

A high frequency curve has significant amplitude fluctuations within short spans, in terms of the visualized area. Sound is an excellent example for high frequency curves. In Figure 9 we display the CDE of the waveform from Beethovens Sym-

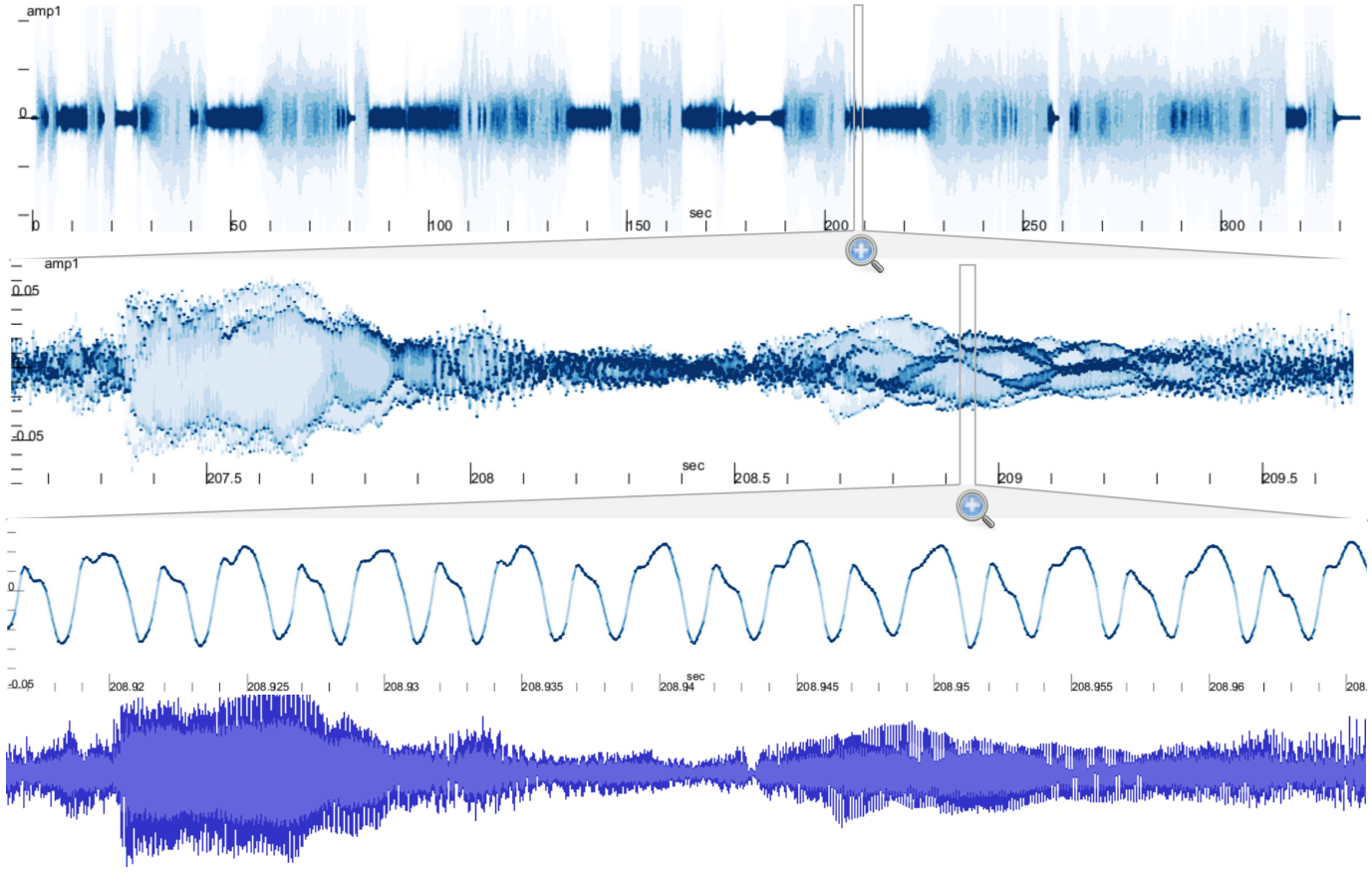


Figure 9: Beethoven, Symphony No. 5 shown using CDE on the full waveform. The top image show five minutes and thirty seconds. The gray box in this top figure shows the timespan zoomed into for the second figure. The second figure shows an interesting feature, spanning three seconds, where a bassoon is playing. The third figure spans 50 milliseconds, zoomed into from the second figure. The bottom figure shows the same span as the second, but using Audacity' viewer [8]

phony No. 5. The top image in this figure displays five minutes and 34 seconds, with almost 15 million samples. We use this example specifically, to show our techniques independence from zoom level. Where the top of Figure 9 showed over five minutes, the next zoom level, in the second graph, spans three seconds. In the third graph, we can directly see the curve, as this graph spans 50 milliseconds. We show this as three discrete zoom levels in this figure, but while interacting with the application, this zooming action is both seamless and smooth. The second and the bottom graph, in Figure 9, both show the same timespan of an interesting piece where a bassoon makes an intricate pattern. This pattern is however, completely lost in the bottom graph, which is the default waveform viewer in Audacity [8]. This intricate pattern is made, in part, by the curve seen in the third zoom level. In this zoom level we see that the curve has two distinct repeating peaks, one with a single mode and the other with two modes, and it is these modes that make up the intricate pattern.

5.2 Prediction Curves

Alan Cox once said:

> I figure lots of predictions is best. People will forget the ones I get wrong and marvel over the rest.

which, somehow, nicely fit the scheme on how modern weather forecasts are done. Instead of a single prediction, an ensemble of possible futures are outlined. However, when a forecast is prepared for public display, it is most often reduced to a single, most likely outcome. When the ensemble of curves spread out, forming a normal distributed pattern, the mode, can correctly be presented as the likely outcome, and the variance, can be presented as the prediction certainty. However, when the ensemble spreads out with two modes, as shown in Figure 5, this model breaks down. We suggest two different use cases for CDE in prediction. In the first case, real-time data is combined with prediction ensemble curves. The historical data will appear as a solid line, and at the most recent sample, an ensemble of curves will possibly spread out, defining the density estimator of the future outcome. This solid line, representing the measured observations is shown in Figure 5, where at $x = 5$ the CDE spreads out into the distribution of future outcomes. By using the full ensemble, rather than the best represented outcome, the operator can also prepare for worst case scenarios, if their probability reaches a given threshold.

The next use case for prediction is repeating, or cyclic patterns. We can find one such cyclic pattern in the yearly temperature. In Figure 10 we show temperature readings, per hour for ten full years by the weather station at Flesland in Bergen, Norway. Data courtesy of eKlima [89]. The temperatures are drawn as ten overlapping curves, using Microsoft Excel in the lower graph. In the middle graph, the moving average over the temperature for all years, and its standard

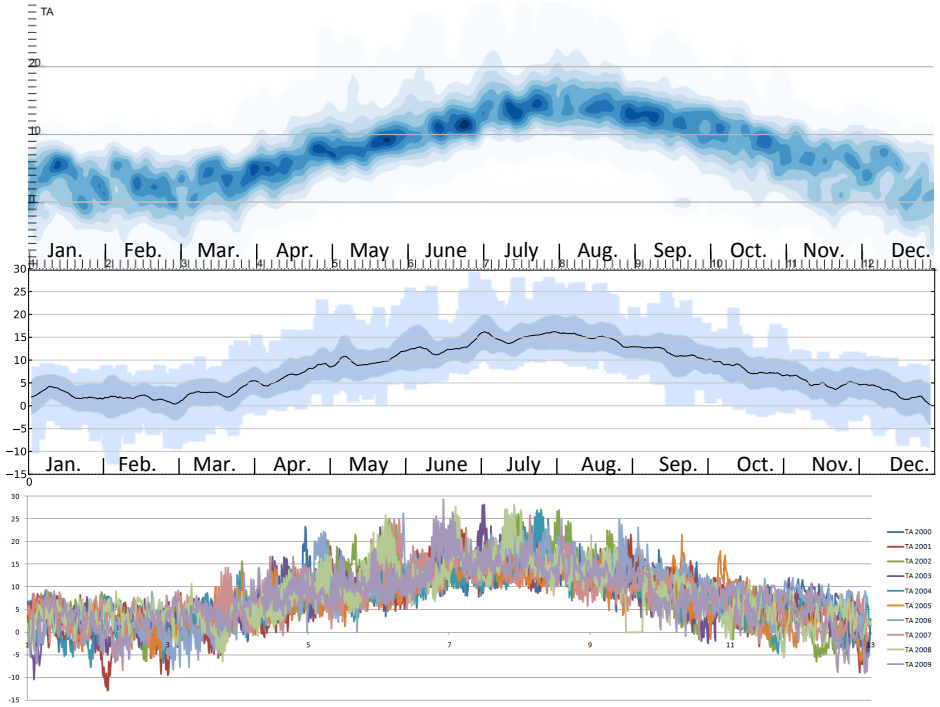


Figure 10: Temperature readings by station nr. 50500 at Flesland, Norway, years 2000 through 2009 with month on the x axis. The top image using CDE, the middle image smoothing using a moving mean and standard deviation, and output by Microsoft Excel™ on the bottom. Notice that the uncertainty/spread of temperature is greater in the winter months Nov. to Feb., than the rest of the year, shown clearly in the CDE. June contains the most stable temperature, represented as the high density there.

deviation is aggregated and shown. The top graph show the CDE for these ten years. By choosing a specific date, the vertical column there represent the probability of which temperatures are likely at that date (according to history). For example, in early June we can see that the probability for the temperature for the EuroVis event is spread out from seven to approximate 18 degrees (given both night and day temperatures). Another way to interpret this CDE, is by looking at the intensity of the highest mode. The higher the mode, the more stable temperature, and vice versa. Historically the temperature is more unstable (less likely to predict a correct outcome) in the winter months, Nov. through Feb. One interesting finding here is the disparity between the mean, as shown in the second graph in Figure 10 and the CDE, for Nov. and Dec., probably due to the less normality of the distribution here.

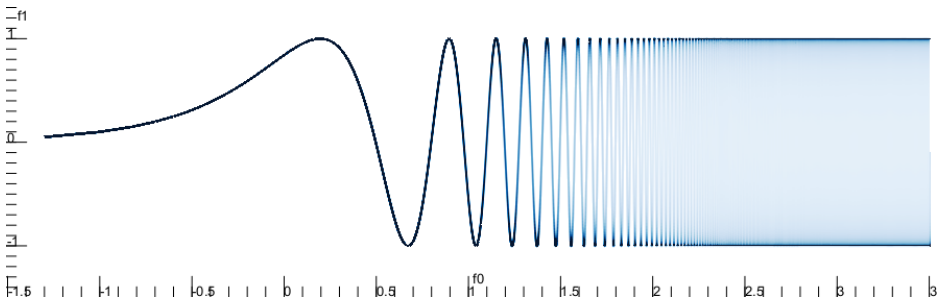


Figure 11: By compressing time with a semi-logarithmic scale, a high level of detail can be read out on recent values, while an overview is available. The logarithmic exponent is given on the x axis.

5.3 Process Visualization

In process visualization, the priorities are often first placed on understanding the *now*, the current situation, followed by both prediction and understanding the historical data. A visualization that receives streaming data, should both emphasize the most recent data, while providing an overview of historical data. Streaming data are, often shown in a temporal window with the most recent values in one end, and the historical data towards the other end. A suitable temporal window is selected, which must be sufficiently small to see the current values, and then data older than this are removed. Figure 11 shows an example use of our CDE with non-linear time, which serves both to emphasize the recent values, to the left, and provide a historical overview that fades into an aggregated probability density estimate.

In drilling, as in many other processes that produce data, we find several data sources that produce bi or multi-modal data. In Figure 12 we show one such example, where the hook-load over time is showed. This image resembles that in our previous work [28], but here it displays time over hook-load, while the other displayed depth over hook-load. Hook load is measured in tonnes, and behaves in a bimodal fashion because the hook is either lifting the entire drill string, or when the drill string is attached in slips to the platform, is zero (actually the weight of the hook itself approx. 40 tonnes). This figure shows the progress over six hours, and we can quickly read out that most of the time has been spent with the drill string attached to the slips, since the mode is highest at 40 tonnes. The second finding, is the spans where the hook load was only at 40, meaning that the operation stalled, and precious time was lost.

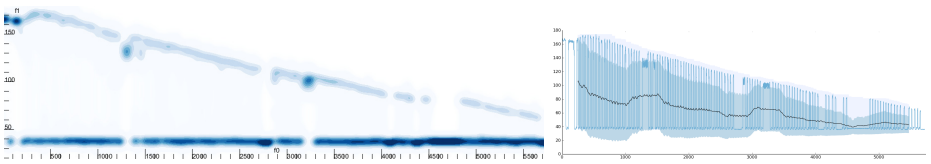


Figure 12: Process data from a drilling operation showing hook-load in tonnes over time in seconds. The right view shows the moving mean, standard deviation and extent, which for this bimodal distribution works particularly bad. The left view displays the curve density estimate of the same data.

6 Summary and Conclusions

We have described the need for a visualization that can represent curves independent of frequencies, zoom level and models, which does not yet exist in the current state of the art. We provide a novel technique on how to render curves independent of sampling-rate, zoom level and curve frequencies. Since kernel density estimates does not impose any model on the distribution, our solution will correctly display data with a single mode, bimodal, tri-modal, or indeed with any underlying model. We have provided implementation details, to promote the usage of this technique in both interactive and real-time settings. Furthermore we have provided several compelling examples of real world usage, showing both where this technique can improve current usages of visualization, but also the versatility of this as a general technique.

For future work, we intend to see how we can use the described technique to provide aid in modeling of data, providing immediate feedback on model suggestions. Furthermore we plan to apply this technique into the daily usage for process visualization, and establish its performance with a user study.

7 Acknowledgements

The work presented here is a part of the project “e-Centre Laboratory for Automated Drilling Processes” (eLAD), participated by International Research Institute of Stavanger, Christian Michelsen Research and Institute for Energy Technology. The eLAD project is funded by grants from the Research Council of Norway (Petromaks Project 176018/S30, 2007-2011), StatoilHydro ASA and ConocoPhillips Norway.

Paper C

Curve-Centric Volume Reformation for Comparative Visualization

Ove Daae Lampe[1,3], Carlos Correa[2],
Kwan-Liu Ma[2], Helwig Hauser[3]

[1]Christian Michelsen Research, Bergen, Norway, www.cmr.no
[2]University California, Davis
[3]Department of Informatics, University of Bergen, Norway, www.ii.UiB.no/vis

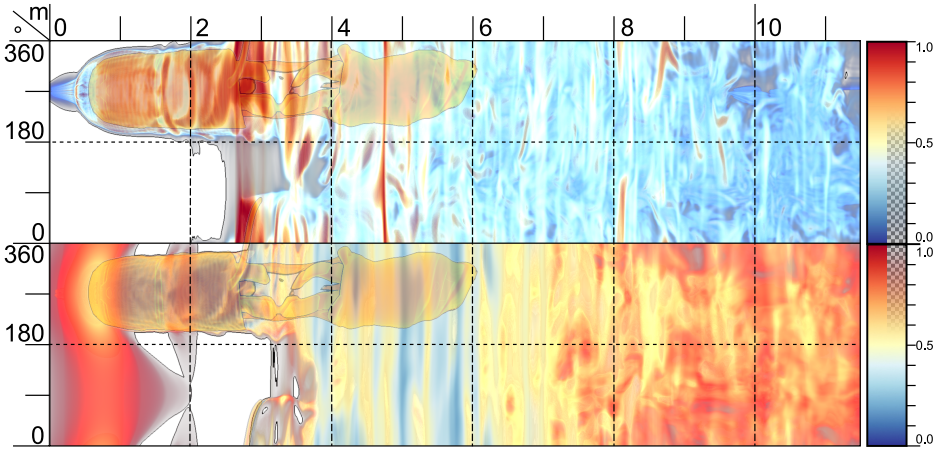


Figure 1: The vorticity magnitude on the top and the velocity magnitude on the bottom (volume-rendered in both cases) of a wind simulation around a car as seen from the inside of a streamline and out radially. The horizontal axis is arc-length of the streamline in meter.

Abstract

We present two visualization techniques for curve-centric volume reformation with the aim to create compelling comparative vi-

This article was published in *IEEE Trans. Visualization and Computer Graphics*, 15(6):1235–1242, 2009 and presented at VisWeek 2009 in Atlantic City by Ove Daae Lampe

sualizations. A *curve-centric volume reformation* deforms a volume, with regards to a curve in space, to create a new space in which the curve evaluates to zero in two dimensions and spans its arc-length in the third. The volume surrounding the curve is deformed such that spatial neighborhood to the curve is preserved. The result of the curve-centric reformation produces images where one axis is aligned to arc-length, and thus allows researchers and practitioners to apply their arc-length parameterized data visualizations in parallel for comparison. Furthermore we show that when visualizing dense data, our technique provides an inside out projection, from the curve and out into the volume, which allows for inspection what is around the curve. Finally we demonstrate the usefulness of our techniques in the context of two application cases. We show that existing data visualizations of arc-length parameterized data can be enhanced by using our techniques, in addition to creating a new view and perspective on volumetric data around curves. Additionally we show how volumetric data can be brought into plotting environments that allow precise readouts. In the first case we inspect streamlines in a flow field around a car, and in the second we inspect seismic volumes and well logs from drilling.

1 Introduction

Successful comparative visualizations build upon one or several shared axes as a reference for attributes that should be compared. Tufte called this visual parallels: "Spatial parallelism takes advantage of our notable capacity to compare and reason about multiple images that appear simultaneously within our eyespan" [122]. Many 2D and 3D visualizations often provide a common reference where comparison is readily available. However, there are numerous cases when the quantities being visualized do not conform to a single shared axis or the common reference makes their comparison difficult. For example, production wells, into oil reservoirs, are drilled with complex geometries and turns rather than, previously common, straight vertical wells. In current operations, most preparations, i.e. well planning, etc., are done in 3D environments, whereas the end product, the drill plan, and all drilling data, is produced in 1D, along well length. Current data analysis with regards to measurements from the well, is done in regular graphs along well-length, but this technique is lacking the spatial 3D context, something we address in this paper. Furthermore, the comparison between two wells of disparate shape and length is difficult in the shared 3D space. Instead, it becomes apparent that a more meaningful comparison is obtained when each well is straightened and the visualized quantities are visualized along their arc-length. Multiple wells of varying length and shape can now be contrasted in a single shared space. A similar need emerges for the comparison of streamlines in a flow simulation. Since individual streamlines can have an arbitrary shape, quantities such as velocity magnitude and vorticity along the streamline, etc., they cannot be compared directly in 3D space due to inter-occlusion and differences in curvature and length. Instead, we can generate a shared frame of reference that straightens each streamline and lets us visualize a relevant quantity in terms of their arc-length. An example is shown in Figure 1, where we display the vorticity and velocity magnitude of a wind simulation around a car in a 2D plane representing the space around a streamline. The visualization maps the complex 3D shape around a streamline into a 2D view that plots the distribution of vorticity and velocity radially around a given point. With this new type of visualization, one can easily *quantify* changes of a certain variable along a streamline and correlate them to arc-length or radial angle. Furthermore, one can correlate the behavior of a quantity among several streamlines.

To achieve these visualizations, we present a general notion of a *curve-centric reformation*, which maps the space around a 3D curve onto a frame of reference relating to the properties of the curve. We present two forms of such a reformation. In the more general sense of reformation, curve-centric volume reformation is a mapping from the original 3D space to a 3D curve-aligned space, where one axis represents the length of the curve, and the other two are the (adjusted) normal and binormal vectors. Unlike traditional visualizations, where objects and

lines are shown in a given 3D space, curve-centric visualizations depict the given space in the frame of reference defined by the curve.

Another type of curve-centric reformation is *curve-centric radial raycasting*, which defines the mapping to a 2D plane, where one axis (in this paper always the horizontal one) represents the arc-length of the curve and the other axis (here the vertical one) represents the cylindrical angle around the curve. This type of reformation is reminiscent of 3D flattening used for the visualization of virtual colonoscopies [12, 58]. Unlike virtual colon flattening, where shapes and angles are preserved for better diagnostics, our radial raycasting approach preserves distances, essential for a meaningful *quantitative comparison* of variables along the curve. This radial ray-casting therefore produces images of 3D volumes from a novel *inside-looking-out* perspective. As the aim of both these reformations is to accurately portray the neighborhood of the curve along arc-length, their intended use is not directly comparable with existing space deformation techniques, that are designed to create alternative (deformed) views of objects.

In our approach, we use a variation of the well known Frenet frame [43] for creating moving frames and provide an implementation that fits into a general GPU-based visualization system. Furthermore we demonstrate the usefulness of our approach in two scenarios. In one, we use a curve-centric reformation to visualize quantities along the arc-length of log wells for oil exploration. In the second, we visualize the vorticity and velocity magnitude of a wind flow simulation around a car. We show that shape reformation provides hints about the smoothness and curvature of streamlines. These quantities, which can be cumbersome to represent in the 3D view at the same time, can now be provided as a comparative visualization.

We make the following contributions: (1) We present a curve-centric deformation of volume data for the purposes of cross-comparison and easier quantitative analysis. As such, our deformation preserves arc length and orthogonal distances from the center. This is a departure from traditional curve-guided deformations which preserve local shape but not distances. Although useful for generating new views of an object, traditional deformation does not ensure the preservation of quantities essential for meaningful comparison. (2) We present a novel raycasting view that provides unprecedented inside-looking-out views of complex volume data. Current approaches for raycasting of deformed volumes exploit the programmability of contemporary GPUs, but the resulting visualizations remain essentially outside-looking-in views of the data. Our results provide novel views that show quantities of interest along important curves.

2 Related Work

Curved planar reformation. An important issue in visualization is the rendering of complex objects in simpler spaces. Some of these issues stem from the

need to compare and measure areas and distances in a 2D plane rather than an arbitrary 3D shape. One such example is curved planar reformation, which maps a volumetric space around a curve to a plane. This idea has been used for virtual colonoscopy and the visualization of other curved structures. Vilanova et al. use nonlinear raycasting to flatten the internal view of a virtual colon [12]. Similar techniques have been proposed by Wang et al. [131], who unravel the colon using a physics-based deformation of the centerline, and Hong et al. [58] and Haker et al. [51], who use a conformal mapping [58]. These approaches use nonlinear raycasting to compute a 2D inside-looking-out image of the colon. A recent approach by Williams et al. [137] also unfolds the colon using multiplanar reformation. However, their visualization is not a planar mapping, but instead an orthogonal projection of the reformed volume. As such, although less effective as a compact map, the result appears more familiar than the planar warping.

In a similar way to colon flattening, Kanitsar et al. presented (and later improved) curved planar reformation for vascular structures [68, 69]. Unlike the colon, vascular structures exhibit frequent bifurcations, which leads to 2D mappings that branch out along with the vascular structures. The reformation of vascular structures was also explored by He et al. [56], who automate the definition of curves by extracting the medial axis of vessels of interest. Lee and Rasch improved this method by considering topological invariant transformations that lead to better visualizations with little artifacts due to reformation [79]. Curve planar reformation also benefits the visualization of misaligned features. For example, Vrtovec et al. [128] uses curve planar reformation to align the central curve of the spine with the sagittal and coronal planes of the 3D images of the spine. This alignment lets radiologists compare different vertebrae in a single image.

In this paper, we are not bound to a particular mapping of a 3D volume, but rather present a more general notion of reformation, called *curve-centric reformation*, which maps the space around a curve to either a 3D volume or to a 2D plane. This enables us to create novel inside-out visualizations of complex datasets, somewhat similar to planar mappings, from oil well exploration to vehicle design.

Space warping. Curve-centric reformation can be also understood as a type of space warping. Space warping is a general methodology for deforming complex objects by warping the space surrounding them. Because volumetric models often have no explicit geometry, this method is often associated to volume deformation. Some of the first attempts to use space warping to deform objects include Barr's global and local deformations, defined procedurally as geometric transformations [11], and Sederberg's free-form deformation [105], which deforms solid geometric models by warping a tri-cubic lattice enclosing that object. To overcome the need for possible dense control lattices, Sumner et al. [113] uses a graph structure to deform the local space surrounding a number of nodes in an object. Singh explores the use of domain curves or *wires* to deform the space

near them [110]. For a complete survey in space deformation, refer to Gain et al.'s paper [46]. Unlike curve-oriented space deformation, often oriented towards obtaining new poses of a geometric model from a set of user defined curves, our curves are not a means for defining deformation but a centerline along which the user can visualize a certain quantity and the surrounding space.

The idea of space deformation was later continued by True and Hughe [118] for volume warping. More recent volume deformations exploit hardware acceleration to obtain volume deformations in real-time, combining control lattices and volume rendering [133, 99]. As an alternative to proxy-based deformation, one can attain the same results by warping the rays used for volume rendering. Nonlinear ray tracing, for example, as proposed by Gröller [49] enables the rendering of nonlinear spaces such as the visualization of relativistic effects, geometric behavior of dynamical nonlinear systems and visualizing particles in a force field. Continuing this work, Löffelmann et al. generalized this technique on how to define a more abstract camera [81], for use in raycasters. Kurzion and Yagel proposed ray deflectors to accomplish volume deformation, which uses point sources to bend rays as they are traced into the volume [78]. Chen et al. generalize this notion to discontinuous deformations in the form of spatial transfer functions [23]. With the advent of fully programmable GPUs, volume deformation has been embedded directly into the raycasting process, enabling the creation of visualizations that resemble surgical illustrations [25]. Deformed volume raycasting, however, retains the outside-looking-in view common in volume rendering. In our paper, we propose novel inside-looking-out curve-centric raycasting views. For a more extensive description of these and other volume deformation techniques, see Chen et al.'s survey [22].

3 Theory

In this section we first present the basis for our approach, in Section 3.1, which is to create a moving coordinate frame, or a tensor consisting of orthonormal vectors. When this foundation is laid, we continue to present our two different curve-centric reformation techniques, in Section 3.2 and 3.3 before we investigate how we can utilize these in comparative visualizations, in 3.4.

3.1 Moving Coordinate Frames

Creating curve-centric volume reformations relates very closely to the problem of creating a local coordinate frame for every point along a curve. This coordinate frame is a tensor, or in other words, a set of orthonormal vectors for every point on a curve $\mathbf{r}(t)$. There are several techniques generating this frame basis for curves, and what separates them is the different problems they solve. The Frenet frame [43] uses the first and second derivate to create a frame that is

intuitive and gives good geometric insight into the curve itself. This tensor is defined by the curve alone, and leaves no control to the user in modifying the tensor without changing the path of the curve. The Frenet frame is however, not defined for curves where the second derivative is zero, e.g. straight sections, or change of curvature. On positions where the sign of the second derivative changes, the tensor "flips". Klok [75] introduced a solution to these sign changes, by introducing a fixed up-vector, and by restricting the curve to those residing in a plane. If we employ Klok's technique on curves in 3D (even though they were not intended for this), the tensor would collapse when the curves derivative is parallel with the selected up vector, and potentially have a sign change ("flip") afterwards. Another method, that can be used to generate frame tensors, is thin plate splines, as introduced by Duchon [32]. With the analogy of bending sheets of metal, this technique produces tensors along the curve where the radial rotation is minimized. This method produces smooth tensors with no sign changes, but this method leaves, similar to the Frenet frame, no control over the direction of the tensor. In our applications we see that preserving a logical up-vector is producing a better spatial reference. Additionally, since the thin plate splines optimizes the global minimum, minor changes to the curve might radically change the result. Because of this, we introduce a technique that allows the user to specify an up vector, while not exhibiting any sign changes. Our technique is similar to that of Klok [75], but allows curves in 3D and the complete tensor is smoothed to avoid sign changes and high frequency changes in the tangent.

The Frenet frame defines the tensor using local derivatives [43]. Let $L \in \mathbb{R}$ be a positive value and $\mathbf{r}(t)$ a parametric curve that is defined for the interval $t \in [0, L]$. The Frenet frame then defines its axes as

$$\mathbf{T}(t) = \frac{\mathbf{r}'(t)}{||\mathbf{r}'(t)||}, \quad \mathbf{N}(t) = \frac{\mathbf{T}'(t)}{||\mathbf{T}'(t)||}, \quad \mathbf{B}(t) = \mathbf{T}(t) \times \mathbf{N}(t)$$

with $\mathbf{T}(t)$ being the unit tangent, $\mathbf{N}(t)$ the unit normal, and $\mathbf{B}(t)$ the unit binormal. The Frenet frame is quite elegant as it can be explicitly computed for every point on the trajectory, but restricts the selection of curves to only those that are twice continuously differentiable, e.g., no curves with points of inflection. Klok's modified Frenet frame [75] is defined for curves in a plane where the normal $\mathbf{m}$ of the plane defines the binormal $\mathbf{B}_K(t) = \mathbf{m}$, and the tangent of the curve $\mathbf{T}(t)$ and completes the frame by $\mathbf{N}_K(t) = \mathbf{B}_K(t) \times \mathbf{T}(t)$. The Frenet frame, Klok's modified Frenet frame, and thin plate splines, place the tangent component of the vector, $\mathbf{T}(t)$, equal to that of the curve's tangent. Figure 2 shows a 2D version of a straightened curve in two versions, the upper using the tangent as a basis for creating the normal, and the lower using a constant normal. In this example this makes a drastic difference on both the readability and the mathematical complexity of the result. Following this we introduce a smoothing

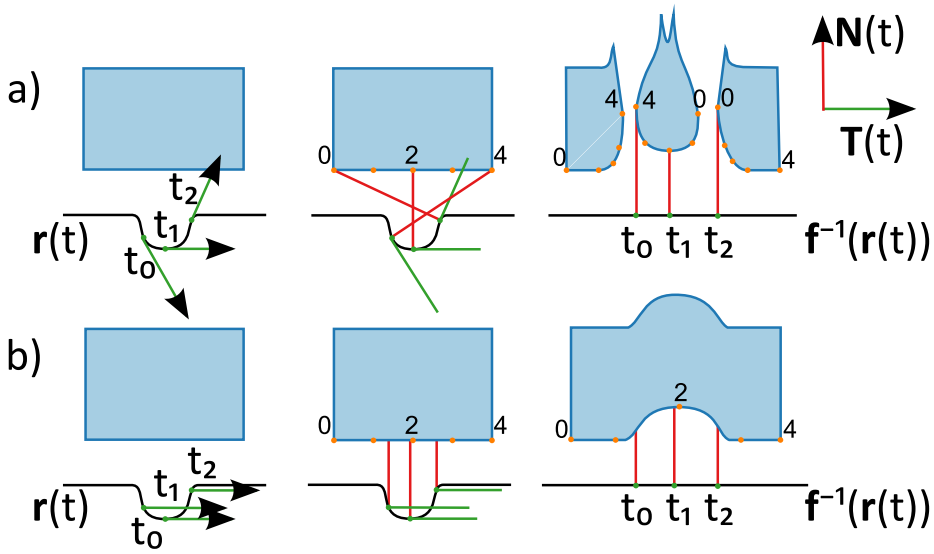


Figure 2: The blue box represents a volume, and the black line the curve to straighten. Two rows showing a) curve-linear deform with tangential $\mathbf{T}(t)$ and b) the same curve-linear deform, but here with constant tangents. Columns left to right show, tangents, normals, and lastly the deformed box.

kernel K_h, with width h. This smoothing kernel is then applied to the original tangent by the following convolution:

$$\hat{\mathbf{T}}(t) \;=\; (\mathbf{T} * K_h)(t) \;=\; \int_0^L \mathbf{T}(\tau) K_h(t-\tau)\, d\tau$$

When applying this convolution to create the smoothed vector $\hat{\mathbf{T}}(t)$ we can experience, with some selections of K_h, a degeneration, where this vector, in a worst case scenario, can evaluate to a zero vector. To avoid this we apply the above convolution/weighted sum after converting the vectors to quaternions.

Another property we would like to translate to the deformed volume is a sense of the up direction. Many datasets have logically defined up/down directions, and in these cases we consider a "desired up vector" $\mathbf{u}$. In all other cases, a consistent vector is chosen instead, as we would like comparable results over several curves. With this vector $\mathbf{u}$, we can now define our *new* normal $\mathbf{N}_m(t)$ and a binormal $\mathbf{B}_m(t)$ according to the rule,

$$\mathbf{B}_m(t) = \frac{\mathbf{u} \times \hat{\mathbf{T}}(t)}{||\mathbf{u} \times \hat{\mathbf{T}}(t)||} \quad \text{and} \quad \mathbf{N}_m(t) = \hat{\mathbf{T}}(t) \times \mathbf{B}_m(t)$$

This rule is very simple, and it is very similar to the one proposed by Klok [75], but unlike his technique this one is not constrained to curves residing in a plane.

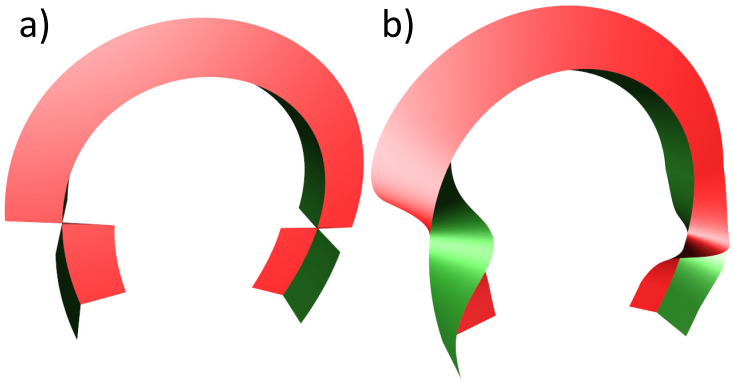


Figure 3: Surfaces following the normal $\mathbf{N}(t)$ as red, and the binormal $\mathbf{B}(t)$ as green. Figure a) shows the modified Frenet frame, experiencing a sign change that the smoothed version, b), handles gracefully.

However, it has two major issues that need to be addressed. The first issue is the obvious case when $\hat{\mathbf{T}} \parallel \mathbf{u}$, and the cross product is 0, accordingly. The second issue is to avoid a sudden sign change of the normal and binormal that can follow such a parallel segment. We solve both these issues by applying a similar smoothing kernel as with $\hat{\mathbf{T}}(t)$. Even more important here than with the tangent, because of previously mentioned sign changes, spherical interpolation must be applied for smoothing, achieved by kernel averaging on the tensors quaternions. From this definition of $\hat{\mathbf{B}}(t)$ we define our normal vector as $\hat{\mathbf{N}}(t) = \hat{\mathbf{T}}(t) \times \hat{\mathbf{B}}(t)$. Figure 3 shows how this smoothing applies to a curve that experiences two sign changes with the modified Frenet frame.

We have now defined a function creating a tensor of orthonormal vectors at each point $\mathbf{r}(t)$ along the curve. In the next section we present two techniques that rely on such a moving frame. To ease the notation and to present that the next techniques are independent of a specific form of frame construction, we will hereby refer to the tangent as $\mathbf{t}(t)$, the normal as $\mathbf{n}(t)$ and the binormal $\mathbf{b}(t)$. However, it is fair to assume that for all practical applications, these vectors are the same as the smoothed versions introduced in this section.

3.2 Curve-Centric Reformation

In curve-centric reformation we aim to create a new volume, parameterized by a curve's arc-length and two terms indicating neighborhood in a distance preserving manner. Unlike several other deformations, the goal of curve-centric reformation is neither to preserve shape, nor angles of external objects, but to correctly portray distances to features and to use these features to provide a spatial frame of reference for the curves trajectory. As shown in the previous section, we make

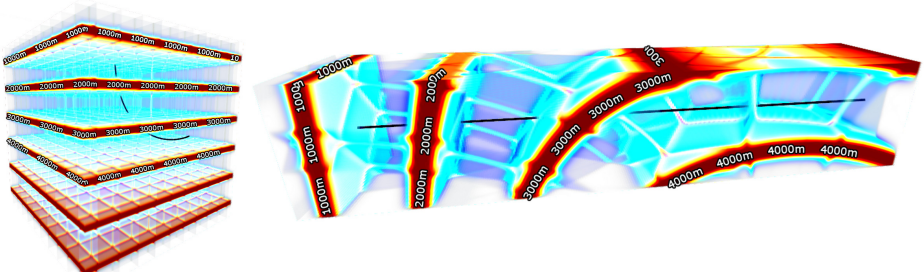


Figure 4: Left, a test volume with a curve in it, and right the result of Curve-Centric reformation. This curve is, after the reformation, the straight line shown in the middle of the volume to the right.

sure that no smoothing is applied to the curve's position, as we would like to strictly enforce these positions as the center of the resulting volume.

We define this reformed volume Y by means of a mapping to the original volume X, or a function $\mathbf{f} : Y \rightarrow X$. Given a curve $\mathbf{r}(t)$ and a defined frame set for this curve, we utilize only the normal $\mathbf{n}(t)$ and the binormal $\mathbf{b}(t)$ in the construction of $\mathbf{f}$. We thus define the inverse transformation, from our reformed volume and back to the original as:

$$\mathbf{f}(x, y, z) = \mathbf{r}(z) + x\,\mathbf{n}(z) + y\,\mathbf{b}(z), \qquad \text{for } z \in [0, L] \tag{1}$$

which satisfies our initial definition of $\mathbf{f}(0, 0, t) = \mathbf{r}(t)$. It can easily be proven that for any curve other than a perfect line, this mapping is not one-to-one, and will usually contain singularities. An important property of this transformation, besides that it preserves the distance of arc-length, is that it preserves distances orthogonally out from the center, i.e., given two points $\mathbf{p}_1, \mathbf{p}_2$ in a given plane orthogonal to Y's z plane:

$$||\mathbf{p}_2 - \mathbf{p}_1|| = ||\mathbf{f}(\mathbf{p}_2) - \mathbf{f}(\mathbf{p}_1)|| \tag{2}$$

the right hand term is, by decomposing $\mathbf{p}_1$ to $[x_1, y_1, z_1]$, and similarly with $\mathbf{p}_2$, equal to:

$$||\,(\mathbf{r}(z_2) + x_2\mathbf{n}(z_2) + y_2\mathbf{b}(z_2)) - (\mathbf{r}(z_1) + x_1\mathbf{n}(z_1) + y_1\mathbf{b}(z_1))\,||$$

which we reduce to, since $z_1 = z_2$:

$$||\mathbf{f}(\mathbf{p}_2) - \mathbf{f}(\mathbf{p}_1)|| = ||(x_2 + y_2) - (x_1 + y_1)||$$

since, our initial condition stated, the two points ($p\mathbf{p}_1$ and $\mathbf{p}_2$) share their z component, the left hand side of Eq. 2 can also trivially be reduced to the same term.

From Eq. 1 it follows that the Z plane denoted by z in Y equals a plane in X defined by the point $\mathbf{r}(z)$ and the tangent vector $\mathbf{t}(z)$. We utilize this planar coherence to create a fast implementation of curve-centric reformation by using *render to 3D texture*, rendering slice by slice fetching samples from X. Rendering these slices z plane after plane, $\mathbf{n}(z)$ and $\mathbf{b}(z)$ is constant per plane, and the evaluation of $\mathbf{f}$ is done after selecting a width w around the curve to straighten, by linear interpolation between four points, $\mathbf{f}(w, w, z)$, $\mathbf{f}(-w, w, z)$, $\mathbf{f}(-w, -w, z)$, and $\mathbf{f}(w, -w, z)$ with sufficiently large w. The result after such a reformation, is a regular 3D texture which can be visualized using, e.g., a regular volume ray-caster as shown in Figure 4.

3.3 Radial Ray-Casting

As presented in the previous section, curve-centric reformation creates a new regular volume, which has to be visualized using a particular method; we now introduce a direct projection of the original volume called radial curve-centric ray casting. Given a function that creates a unit normal $\mathbf{n}(t)$ and a unit binormal $\mathbf{b}(t)$ from a curve defined by $\mathbf{r}(t)$, traversing the arc-length t we cast rays starting at $\mathbf{r}(t)$ in the direction

$$\sin(\phi)\,\mathbf{n}(t) + \cos(\phi)\,\mathbf{b}(t), \phi \in [0, 2\pi]$$

In its simplest form, by using a straight line, this radial ray-casting technique is reduced to a cylindrical projection, which form a projection that lies in between perspective, having a viewpoint, and an orthogonal projection, having a plane of view, with a line as its starting point. Using this technique one creates 2D projections that have no perspective distortion in the horizontal direction, i.e., the direction along the line, but with perspective distortion in the vertical direction, i.e. the angular rotation around the curve. However in addition to the perspective distortion, for curves, other than those of a perfect line, the curvature of this curve, or rather the torsion of its moving frame tensor, will further distort objects and features as seen from the curve.

A common strategy for ray-casting is to first render the texture coordinates of the exit points for rays, and then to store them. Next the ray casting is initiated by rendering the starting location of these rays, and iterating the ray-casted volume using the line between start and exit. This technique is depcited in Fig. 5, where the left part shows a box used as proxy-geometry for rendering the entry and exit buffer. The strength of this technique is its simplicity and that it is independent of perspective and proxy geometry, and as we will show, also compatible with our radial ray-casting. Generating the exit points is usually done by rendering the proxy geometry, culling front faces, but we will use a

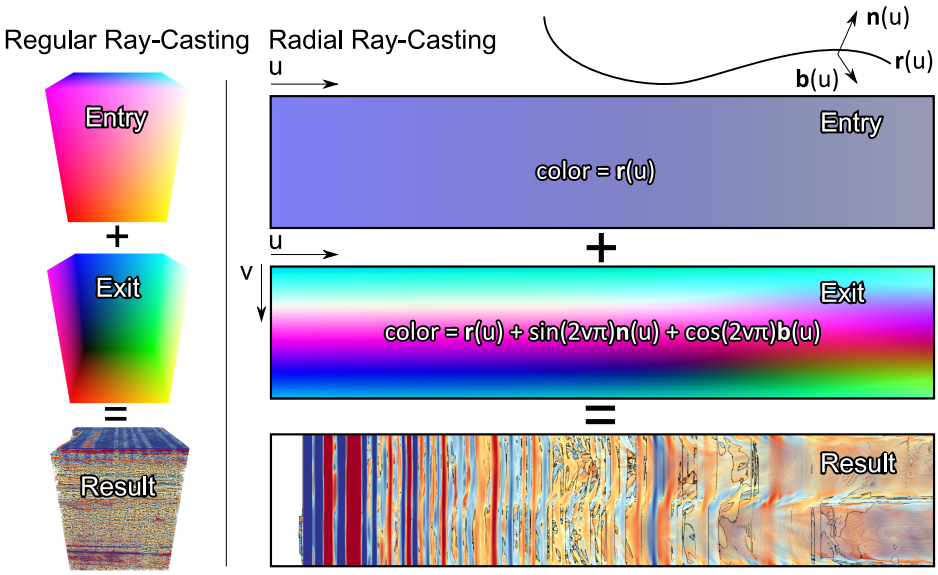


Figure 5: Regular ray-casting uses proxy geometry, generating the entry and exit positions for ray traversal. Our radial ray-casting technique mimics this behavior by setting the entry buffer to the curve's position, and the exit buffer as the curve's position plus an angular rotation around this curve.

more direct method. The exit points in radial ray-casting are defined as, over normalized screen-space u, v, using a far "plane", or tube, with radius far:

$$\mathbf{r}(uL) + far \cdot (\sin(2v\pi)\,\mathbf{n}(uL) + \cos(2v\pi)\,\mathbf{b}(uL))$$

For efficiency concerns, this exit point should also be clipped, at the intersection point to the texture cube, to stay within existing volume coordinates. After this exit buffer is established, we can start our radial ray-casting, using the start position $\mathbf{r}(uL)$. This start/entry and exit buffer is shown, on the right of Fig. 5. Given the exit position buffer and the starting position, this radial ray-casting technique is theoretically compatible with shader code written for any other ray-caster employing the same strategy, and will thus render at real-time speeds, and be able to perform advanced dynamic transfer functions. When applying proven shader techniques to this type of rays, one of course needs to keep in mind the aliasing issue, produced by the fact that the projection will sample the surrounding space with an uneven number of rays, due to the curvature, and the combination of almost parallel rays in the direction of the curve and an extreme perspective, 360 degrees (like in fisheye projections), in the rays orthogonal to the curve. Figure 1 (the teaser image) shows two examples of radial ray-casting, where a regular color map transfer function has been used, in addition to edge

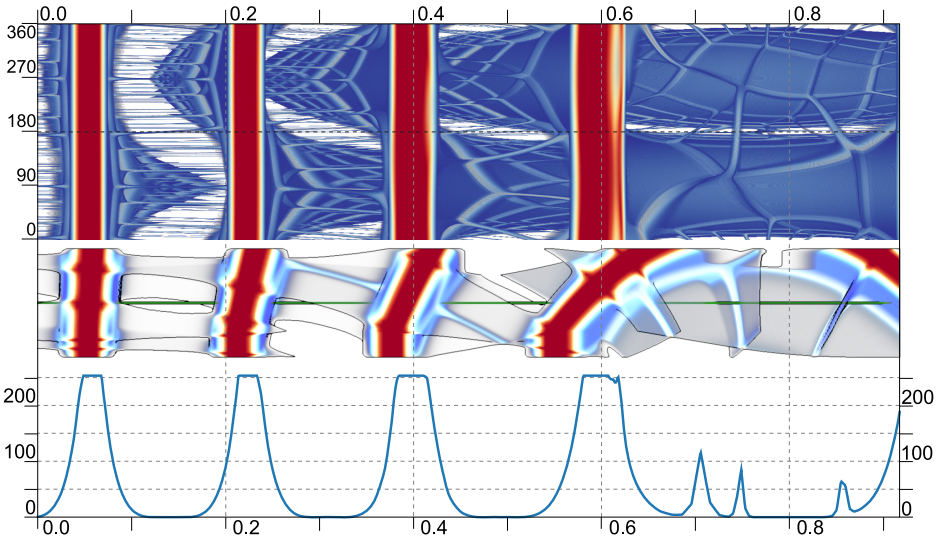


Figure 6: The curve from Figure 4 shown as a curve-centric radial projection, top, as a reformed volume visualized, middle, and a curve plot, showing the sampled intensities from the volume along the curve. This comparative visualization allows accurate comparison of intensity values to their spatial origin.

enhancement. Another example is shown in Figure 6, where the top image shows a radial ray-casting, and because of properties in our generation of the moving frame, we can read out the direction around the curve, as 0 degrees corresponds to the left of the curve, looking in the direction of the trajectory (left to right), 90 degrees up, 180 right and 270 is below the curve.

Unless the curve is dynamic, this rendering produces images with a fixed "view", but one can still implement an interactive exploration of the volume around the curve, by constraining to either a subsegment of L or a subset of the angular radii. An interaction scheme would then allow for zooming, a reduction of segments in both dimensions, and panning, which would translate this segment.

3.4 A Common Axis for Comparative Visualization

Two objects sharing one or more axes have a basis for comparison. A shared axis is simply one that has the same unit, e.g., two physical objects share size, and can thus be compared in terms of size. Moreover, the display of those shared axes should be shown in scale to each other, optimally sharing the exact same scale, to avoid producing deceptive visualizations. In this paper, we present two techniques that create visualizations, by deforming or projecting space, that are aligned with the arc-length of a curve in space. The strong rationale for pursuing

this alignment is to be able to create visualizations that combine well with existing techniques, like graphs, 2D plots, or even images. These are techniques that are not always well imported into a 3D environment, e.g., Figure C.7(a) shows 1D graphs for multiple wells drawn in 3D space, enabling coherence between values and their spatial position, but is not suited for accurate readouts. By reversing this strategy we instead investigate how well we can display information from volumetric models with a correct alignment to the 1D space of these 1D and 2D techniques. One way to look at this reformation is that one renders the physical space in the logging space, instead of rendering the logs in the physical space. A strong argument for this reformation, in Figure C.7(a)'s case, is that the function shown on a single well here is actually only one of several production related parameters that the production team is interested in analyzing. By straightening a single well, all of the different production values can be shown in its own graph, without the inherent occlusion problems that would exist in a 3D visualization. Another argument for exactly this reformation of space, is to provide accuracy in display, e.g., in a plotting environment one has a direct readout of values. Figure 6 shows such a plotting environment, where the two separate displays share the horizontal axis, and one could theoretically use a ruler to match peaks in the bottom curve to features in the two above. The curve-centric radial projection makes sure to preserve a required orthogonality in the direction of the 1D axis of the plot. This orthogonality is also enforced in the visualization of the deformed volume by simply rendering it with an orthogonal projection.

The curve-centric visualization techniques are only fully utilized when actually shown in the same space as important arc-length parameterized metrics. It is not within the scope of this paper to create other novel visualization of these metrics, but to show how one can create comparative visualizations by combining deformed and curve-centric projected, with existing and well established techniques. In the next section we show several examples of arc-length parameterized data, but these are just some of the many that actually exists. As good examples for these data sources, we can imagine most of the sensing devices that are attached to moving objects. These sensing tools are logging data by time, but indirectly they are also portraying information about their surroundings as a function of where they are. Other data sources are, e.g., 1D simulations or trajectory simulations.

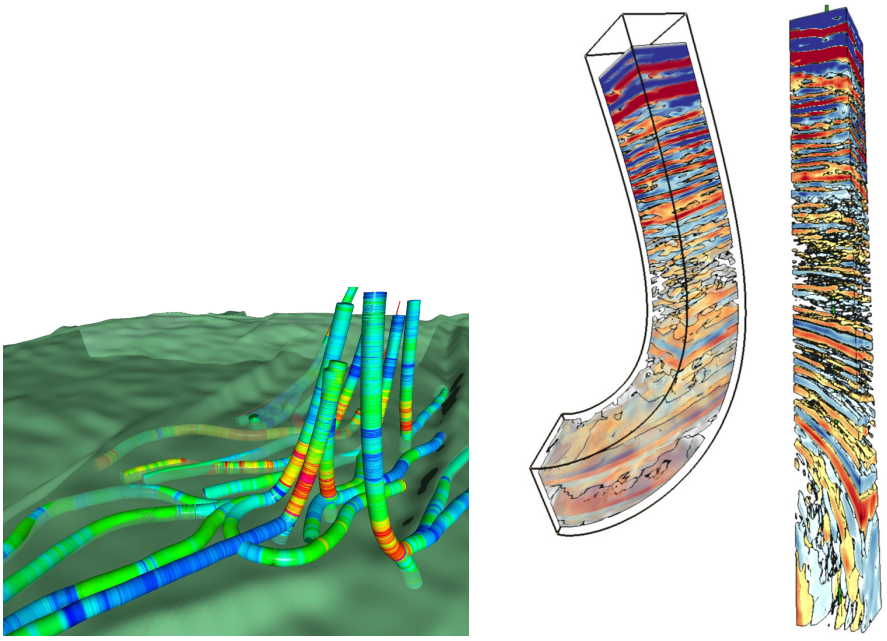
4 Application Cases

In this section we show how we successfully applied our techniques to cases of two different industries. In Section 4.1 we investigate two cases with application to the petroleum industry; the first case where we show how curve-centric reformations can help show the multiple data sources gathered when drilling exploration wells; and in the second, we show its application in a real-time drilling scenario, looking

closer into well-bore uncertainty at a certain strategic depth, motivated by an actual incident. In Section 4.2 we similarly investigate two cases with application to the car industry. In both of these two we investigate parameters around streamlines, but in the first one we investigate multiple parameters of a single streamline, whereas in the second we compare several different streamlines on a single parameter.

4.1 Well-Centric Visualizations for the Petroleum Industry

The petroleum industry uses drilling in part for exploration and in part for creating or expanding production in oil and gas reservoirs. Where vertical wells are almost the norm for exploration wells, an increasing number of horizontal wells and even those with more complex geometry, are used to extend the lifetime of aging reservoirs. While drilling these wells, operators apply several sensing tools to learn about the surrounding formation, to get situational awareness, to predict potential problems, and to maximize the drilling parameters, e.g., rotation speed, how much pressure should be applied to the drilling bit, and more. These sensing tools can measure down-hole pressure, electric conductivity of the formation wall and a number of other physical parameters. These provide output with a timestamp, or streaming data in time, but for many of these physical parameters, different representations make more sense. E.g., a sensing device measuring the physical properties of the rock outside the drill-string will provide a measurement as a function of measured depth (arc-length, not true depth) and to a lesser extent rotational position, rather than time. Similarly there are many other parameters that are best shown as a function of depth rather than time graphs, e.g., rate of penetration, hook-load, electric logs, and more. When these logs, all having the common axis of arc-length, are shown to explore their relation to seismic volume data, existing techniques usually put these logs into the 3D seismic space, as shown in Figure C.7(a). We argue that this technique highlights the seismic volume as the important feature, showing the logs in a contextual manner. We propose to invert this display, and to show the seismic in the space of the logs instead. This is done by a reformation of the volume around the wells, and thus the seismic volume share arc-length as the axis with logs. An example of this is shown in Figure 8 where different measures along the drill-string are compared for coherence, as is a common operation in exploration drilling. This figure shows a volume visualization of the deformed volume on top, containing the well, an orange line, in the middle. The next is a graph of seismic reflectance sampled along the wellpath. As expected we see a strong correlation between this and the one above and below, which is the radial ray-casted volume as seen from the well. The final image, at the bottom, is a physical measured value, called an UBI image, or Ultrasonic Borehole Image. This imaging device produces 360 degree images of the formation around the well, very much like our radial raycaster, and we would thus expect to see a very tight correlation be-



(a) (b)

Figure 7: (a): Multiple production wells shown as tubes, where color indicate a single physical measured variable along the well length. All these wells reside in the reservoir, some injecting fluids while others receiving fluids. Image used with permission by StatoilHydro ASA. (b): The well we investigate in this case. This figure shows the seismic volume in the close vicinity to the well, before and after reformation.

tween these. Initial seismic volumes are often quite uncertain, and to minimize these uncertainties they are compared to results from exploration wells. We see the techniques of curve-centric reformation as a natural addition to the already existing techniques for studying these results.

Real-Time Drilling Data is the second case in which we introduce a novel view on drilling. The contractors that drill wells have to follow a strict plan of where the drill string should be on the way to the target. In the preparation, to create a drill plan, the close proximity to the well is mapped out and simplified into a 1D plan of different properties that the wellbore passes through. These properties are, e.g., pressure gradients, lithology (rock type), and where stops, logs, and casings should be positioned, all of which share the axis of arc-length. A case using some of these drilling properties is shown in figure 9, in which we inspect the properties of the well as shown in Figure C.7(b) before and after reformation. In this case we show an important part of a drilling operation, in which we are drilling the final stretch before we enter the reservoir, denoted by the

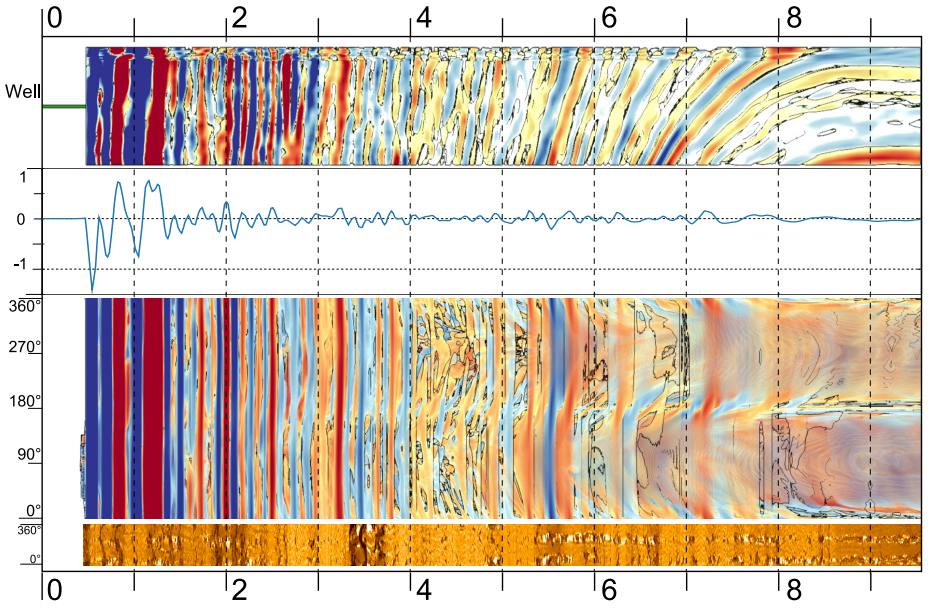


Figure 8: The above images show different measures along a well-path. From top, an orthogonal ray-casted view of a curve-centric deformed seismic, second graph shows the volume intensity (seismic reflectance), third shows an angular ray-casted view from the wellpath and out into the surrounding seismic, and lastly the final image shows an ultrasonic borehole image, an angular view into the formation.

blue limestone. Because of very different properties of the shale (green) and the limestone it is very important to stop drilling as close as possible to this horizon, and insert a casing to protect the wellbore and formation from changing pressures. In addition to providing spatial awareness, this figure highlights an important detail our petroleum industry partners has expressed interest in, namely the wellbore uncertainty. The red ellipse, shown in the same figure (9), represents the area, or rather the volumetric ellipsoid, in which the drill-bit is positioned with 95.4% accuracy. Notice that the 1D lithology column shows the limestone to start at the wellbore's intersection with a major horizon at 5.45 which is diagonal. It is fair to assume that everything below this diagonal horizon is included in the reservoir. This shows that the 1D lithology is only accurate if the wellbore traverses where it was planned to. The 1D lithology column fails to incorporate the uncertainty of wellbore positioning, eg., the wellbore could be positioned deeper, and the reservoir would encountered earlier. In fact, the ellipse shows that given a 95.4% probability safety margin, one *cannot* guarantee that we have not already entered the reservoir. Projecting this ellipsoid into the

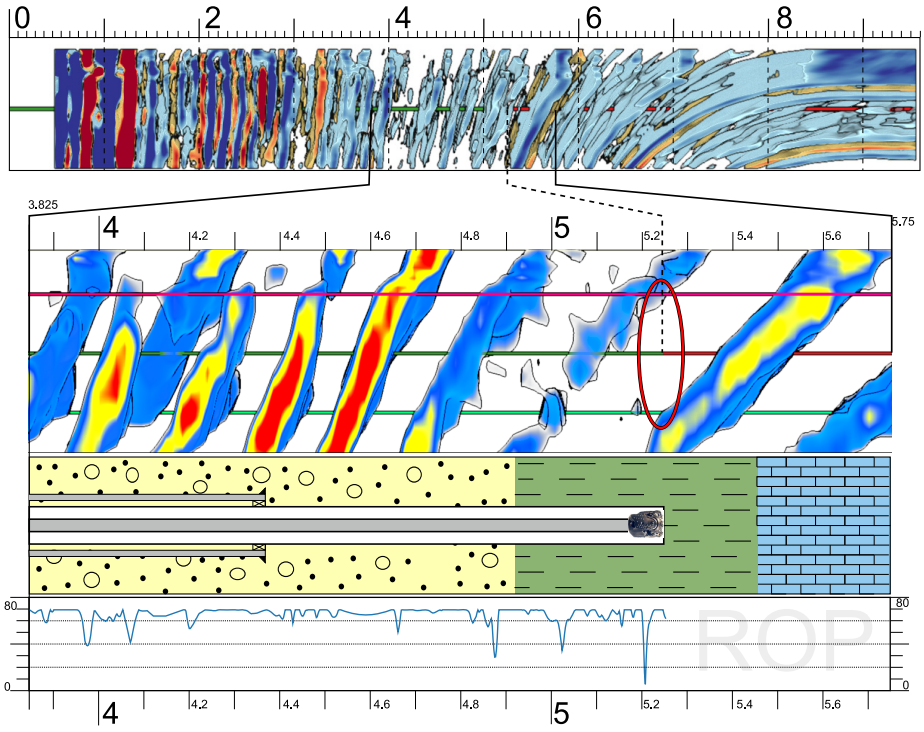


Figure 9: Deformed seismic can provide spatial reference for real time drilling data as well as showing uncertainty in the 1D lithology column. Image on top is the full length deformed wellbore, below a zoom-in of the section currently drilled, which also contains the 1D lithology with the current drill bit position, and a real time graph showing the rate of penetration (ROP) for the section. The red ellipse in the center shows positional uncertainty.

1D lithology column would not reveal this. As mentioned earlier, this example is motivated by an actual incident, where wellbore uncertainty led to the wrong assumption that one had a good clearing before entering the horizon. Secondary measures (gas show) did in fact even show signs that the reservoir was entered, but was ignored due to the believed clearing, and drilling was resumed. We contest that if our display would have been used in this case, then the secondary sign would not have been ignored, since the probability is already shown to be below the threshold for entering the horizon.

4.2 Streamline-Centric Visualization

In this section we investigate a dataset containing velocity and pressure from a single time step in a wind simulation for vehicle design. Additionally we created

a distance field from a geometry file. From the velocity field, we extracted two scalar fields, velocity magnitude and vorticity magnitude. An overview of these datasets are shown in Figure 10. From the original velocity field we further extracted several streamlines, from which we selected a few streamlines that shared an interesting property in that they traverse in close proximity to a side mirror, an overview of these streamlines are shown in Figure 11. Inspecting one of these streamlines is enabled in our shared axis view, where multiple views on the streamline are provided for comparison. This comparative visualization display, as shown in Figure 12, is well suited for understanding how the streamline is affected by the different fields, in addition to looking at the correlations between the different fields. The deformed car on the top of Figure 12 acts as a spatial reference, additionally to reveal information on the curvature, the higher the curvature the bigger the deformation on the car. In this view (still Figure 12) we can see that there is a very good correlation between velocity magnitude and pressure in front of the car. At 1.5 meters into the streamlines arc-length, the streamline passes through a positive pressure, which is aligned with a slight drop in velocity. More interesting is the low pressure the streamline passes through at 2 meters, which interestingly enough does not seem to affect the velocity. Right behind the car, at 5.5 meters, we can see a drop in velocity (the blue vertical feature), that does not seem to correspond with any of the other views. One possible explanation might be the vorticity and low pressure feature right in front of it. Studying correlations in this manner does provide a new perspective into the study of flow fields that our industry partners found intriguing.

Our application partner, a team of engineers who use computational fluid dynamics simulations for all aspects of automobile design, was intrigued by the alternative views that we are able to create with deformation and the curve-centric radial projection. The concept of applying deformation on the car body to reveal a secondary effect outside of the car is very enticing to them. This deformation would then be a tangible communication between the bodywork designers and them for shape optimization. In particular, they pointed out that by choosing the appropriate flow quantities other than velocity, such as vorticity vector or helicity density, the deformed surface could actually suggest the location and extent of the shape change needed to achieve optimal performance. Additionally, the curve-centric radial projection of the flow structure with respect to a straighten field line may display information hard to reveal with conventional flow visualization techniques, such as drag force acting on the car.

5 Summary and Conclusion

In this paper we have presented a general solution on how to create new curve-centric parameterizations of volumetric space. Additionally we have presented how to use this new parameterization in creating two visualizations that are

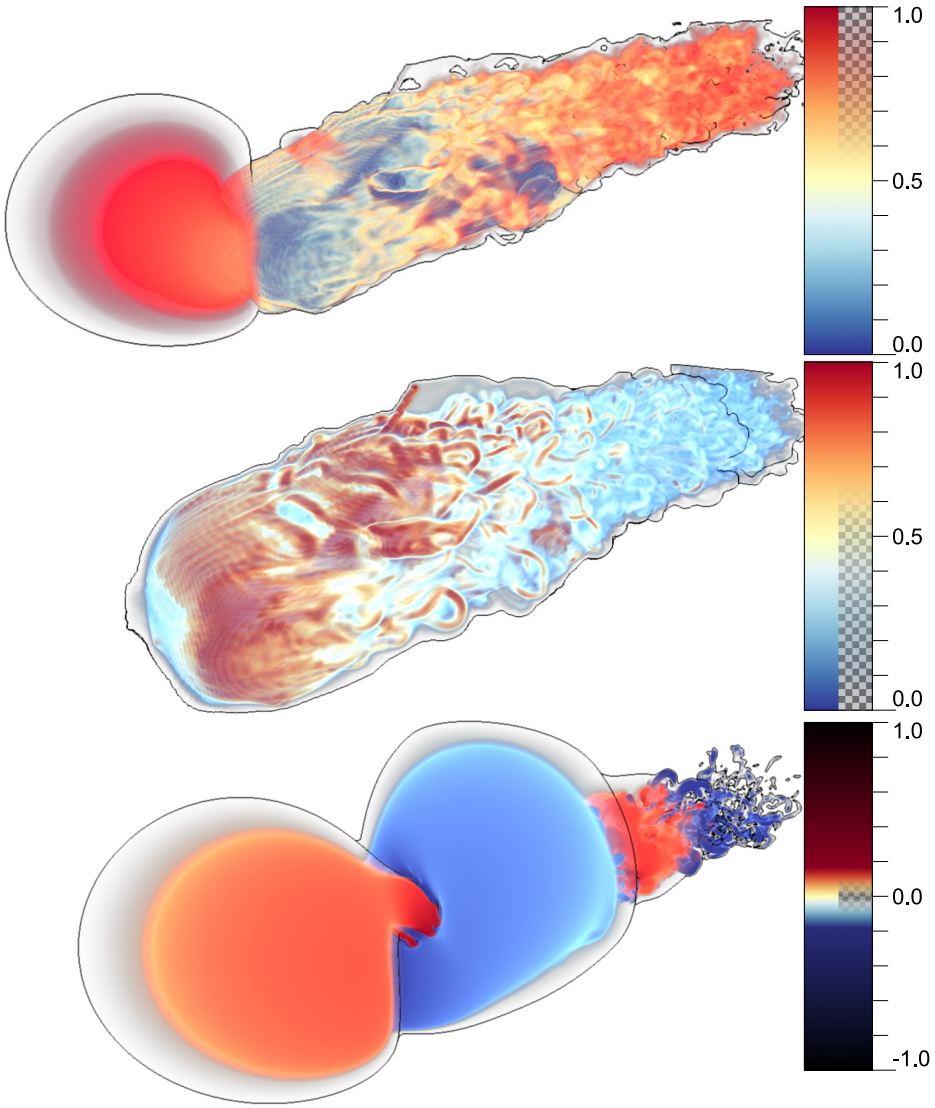


Figure 10: Top, velocity magnitude, middle, vorticity magnitude, and below pressure, all from a vehicle design simulation.

aligned with the arc-length of the curve. We have shown that we can use this alignment to create comparative visualizations, where 3D spatial positions are shown directly overlapping with 1D, or 2D arc-length parameterized functions.

We have successfully created a prototype using a combination of C++ and

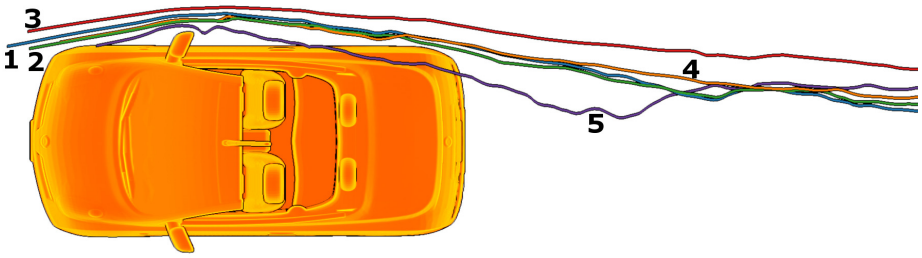


Figure 11: Showing five selected streamlines with a close proximity to the side mirror, that provides interesting features to study.

Python, which implements the creation of a moving frame given a curve, an optional up vector and a smoothing factor. Using this moving frame and supplying a volume, 3D texture or array, and a width, the prototype creates a deformed volume either as a 3D texture or as an array. This prototype supplies methods for creating images rendered either offline or real-time of both the deformed cube, as an outside in view, and the radial 2D projection, giving inside out views. Using this prototype we have created two case specific applications, one to investigate streamlines, and one to investigate well data from the petroleum industry. To avoid reinventing existing visualization techniques we have integrated our techniques with matplotlib: A 2D Graphics Environment as presented by Hunter [59], which enables the use of existing 1D and 2D visualization techniques, along the aligned axis of our result. Our test system has an Intel Core2 Quad CPU and a GeForce 8800 GTX. Creation of a deformed volume with dimensions [128,128,512], from an original volume [600,300,750], takes 19 milliseconds. Rendering a radial ray-casted image with dimensions [2048,512] of the original volume takes 51 milliseconds.

Limitations and Future Work: The presented algorithm requires a user specified up vector and smoothing factor, and for the volume deformation, also a user specified width. While this user input provides flexibility, it does represent a limitation for complete automation. There is a correlation between the frequency of changes of the curve, the width of the box surrounding the curve, and the smoothing factor. This correlation is not explored in this paper, but could potentially give some interesting automation of these parameters. The radial ray-casting technique does not duplicate any of the original data, but the volume deformation does. When straightening curves with high frequency changes, which will then have a large arc-length within a small section of the original volume, this duplication of voxels becomes very apparent. Another limitation, is when sections of the curve, larger than the smoothing kernel, are parallel with the up-vector. We proposed to solve this by modifying the up vector, or by in-

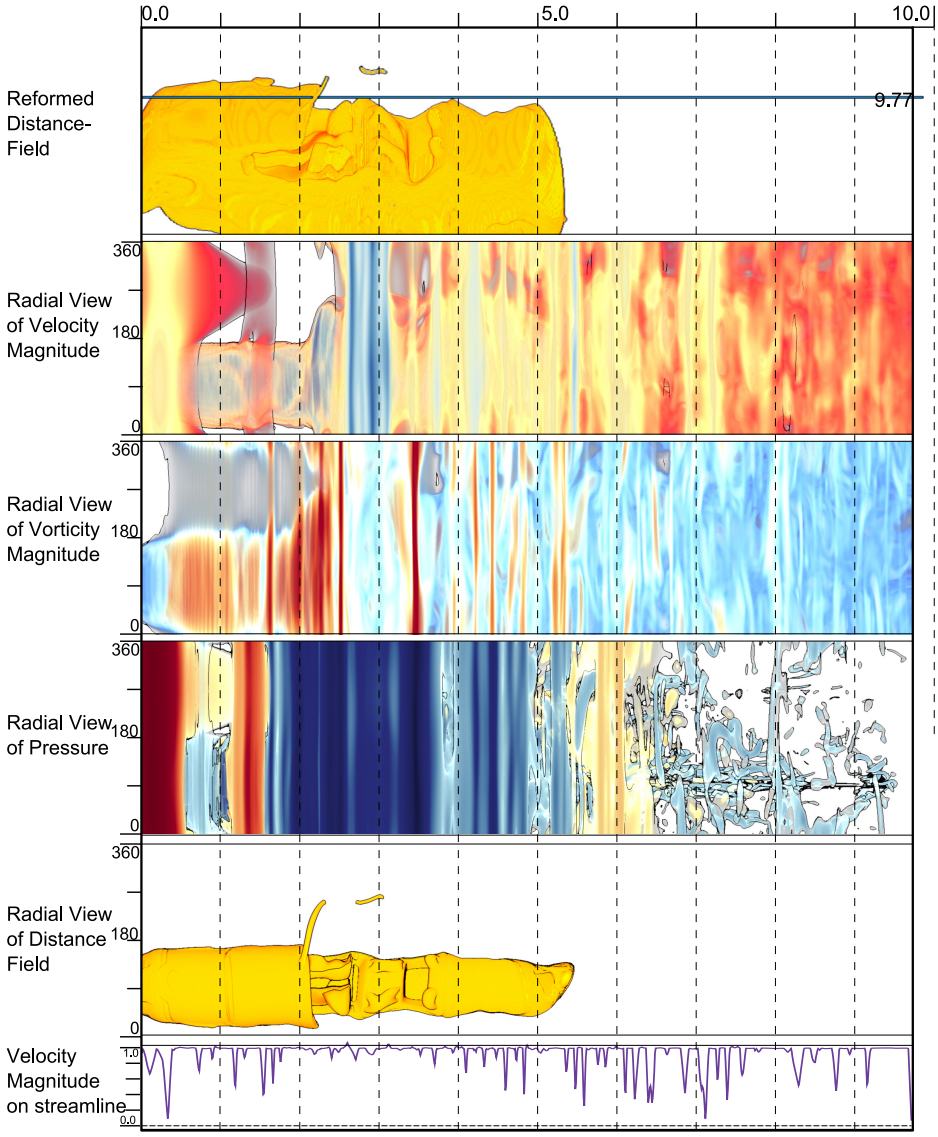


Figure 12: A plot showing streamline 5 as shown in Figure 11. The deformed car is shown on top, which then acts as a spatial reference for the measurements below. The three measurements below are radially ray-casted streamline-centric views, and below a graph tracing a value on the streamline.

creasing the smoothing factor, but the smoothing kernel could be expanded to take these longer sections into account, and smooth over them by using a varying smoothing factor.

In cooperation with our application partners we have used our prototype applications to show how curve-centric visualizations, combined with application specific data, can create effective and compelling comparative visualizations. Moreover, we have produced a more general approach, without the same application specific requirements as shown in previous techniques such as virtual colonoscopy or curve planar reformation, one that ideally could be used for numerous other applications as well.

6 Acknowledgments

The work presented here is a part of the project "e-Centre Laboratory for Automated Drilling Processes" (eLAD), participated by International Research Institute of Stavanger, Christian Michelsen Research and Institute for Energy Technology. The eLAD project is funded by grants from the Research Council of Norway (Petromaks Project 176018/S30, 2007-1010), StatoilHydro ASA and ConocoPhillips Norway. The vehicle simulation data was provided by Dr. Kenji Ono at Riken, Japan. This work was supported in part by the U.S. National Science Foundation. The authors thank StatoilHydro for providing seismic data, wells and well-logs, Chris Ho for providing the streamline extraction code, and finally the reviewers for valuable feedback.

Paper D

Interactive Difference Views for Temporal Trend Discovery in Multivariate Movement Data

Ove Daae Lampe[1,2], Johannes Kehrer[2] and Helwig Hauser[2]

[1]Christian Michelsen Research, Norway, www.cmr.no
[2]Department of Informatics, University of Bergen, Norway, www.ii.UiB.no/vis

Abstract

Movement data consisting of a large number of spatio-temporal agent trajectories is challenging to visualize, especially when all trajectories are attributed with multiple variates. In this paper, we demonstrate the visual exploration of such movement data through the concept of interactive difference views. By reconfiguring the difference views in a fast and flexible way, we enable temporal trend discovery. We are able to analyze large amounts of such movement data through the use of a frequency-based visualization based on kernel density estimates (KDE), where it is also possible to quantify differences in terms of the units of the visualized data. Using the proposed techniques, we show how the user can produce quantifiable movement differences and compare different categorical attributes (such as weekdays, ship-type, or the general wind direction), or a range of a quantitative attribute (such as how two hours' traffic compares to the average). We present results from the exploration of vessel movement data from the Norwegian Coastal Administration, collected by the Automatic Identification System (AIS) coastal tracking. There are many interacting patterns in such movement data, both temporal and other more intricate, such as weather conditions, wave heights, or sunlight. In this work we study these movement patterns, answering specific questions posed by Norwegian Coastal Administration on potential shipping lane optimizations.

This article was published in *Proc. Vision, Modeling, and Visualization (VMV 2010)*, pages 315–322, 2010 and presented at VMV in Siegen, Germany by Ove Daae Lampe

1 Introduction

Massive streams of complex time-dependent data arise in various areas of business, science, and engineering (resulting from large-scale measurements, modeling, or the simulation of dynamic processes). Being able to understand time-related developments allows one "to learn from the past to predict, plan, and build the future" [2]. This can play a major role in scenarios such as the analysis of critical process workflows and developments, project planning or process simulation, and to develop alternative scenarios if required. In our case, the Norwegian Coastal Administration (NCA) has been asked by the Norwegian government to perform an analysis of whether a sea tunnel should be made on Stad. Most of the Norwegian coastline allows vessels to safely travel inshore, protected from the harsh weather of the North Sea by a large number of bigger and smaller islands (see Figure 1). At Stad, however, vessel traffic is forced out in the open sea. This presents a problem since there are demanding wave conditions 90 to 110 days a year in this area.

The tunnel in question would traverse underneath the peninsula below Stad near Selje, and be 1.8 km long, 23 meters wide, 45 m high including 12 m water depth, producing an excavated mass equal to 3/4 of the Giza pyramid. Building this tunnel would amount to a large national endeavor due to its size, so a careful economic rationale is needed in the first place. Part of the rationale would consist of a decreased risk for the vessel traffic in this area. Another one would be saved costs by having vessels not needing to wait for good weather north and south of Stad. The questions of interest for NCA therefore were, how significant the correlation of waiting periods and bad weather is, and whether we can quantify the amount of (lost) hours that could be saved by having the tunnel as a weather-safe short-cut. Another line of questions addresses the potential risk reduction when having such a tunnel. Accordingly, we were interested in how the weather affects the vessels' choice of paths, e.g., do they go closer to shore, or not, when the weather is bad.

Faced with these questions, we engaged with an analysis of the Automatic Identification System (AIS) data for a large historical collection of vessel movements. AIS is a radio system, broadcasting vessel ID and position at regular intervals, that all vessels above a certain size must have in these waters. Coupled with historical weather observations from stations in the vicinity of Stad, we should have the required data to answer these questions. According to Andrienko and Andrienko [5], AIS data contains three attributes characterizing agent movement data, i.e., agent identifier, time, and spatial position. Selecting all data by identifier, and ordering it by time, makes a trajectory, and many of these trajectories then makes a movement dataset. Furthermore, AIS contains a varying number of attributes per vessel, e.g., vessel length, vessel type, and nation, and attributes per journey such as persons on board, destination, and cargo. When we further extend this dataset by spatiotemporal attributes, such as wind direction, wind

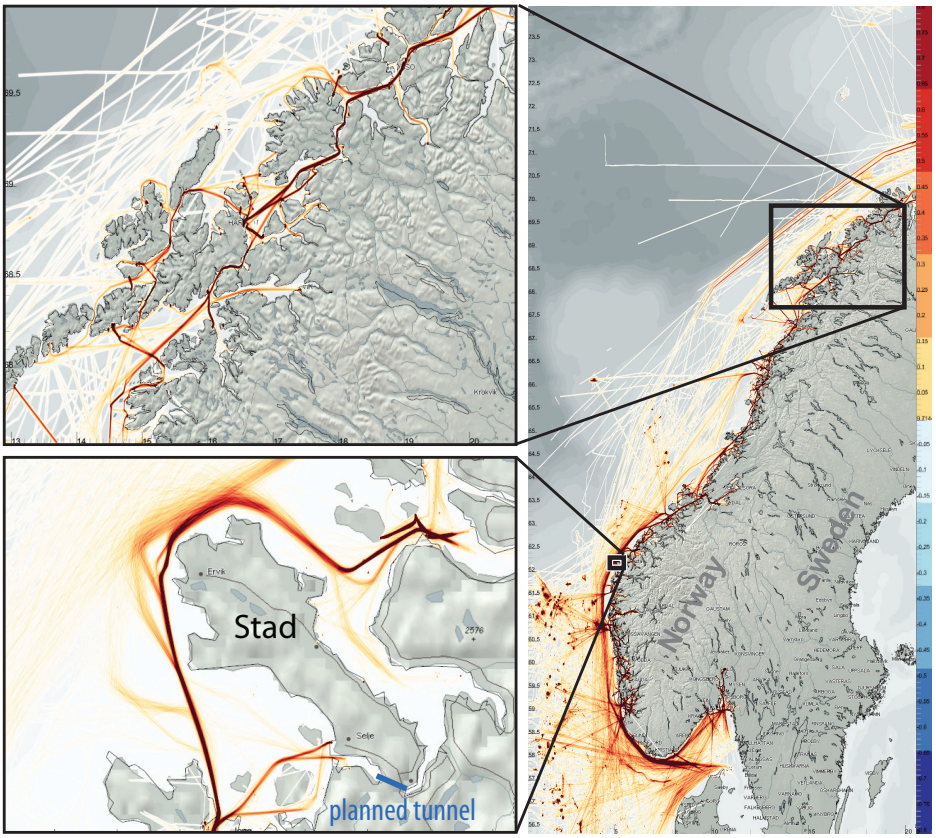


Figure 1: Vessel movements around the coast of Norway. At Stad (lower inset) the traffic is forced into the open sea, while usually most (local) traffic is within the outer islands protected from the weather (exemplified by the top inset).

speed, and wave height, we indeed have a multivariate movement data visualization challenge at hand. Another consideration that we have to take into account is that the bigger the datasets get, the better the statistical confidence of our findings can get. This means that if we have sufficiently many trajectories, we can consider the data as a probability density estimation, as opposed to a set of just a few samples. AIS is collected by a huge network of radio transponders along the coast; on the other end, it is also dependent on the transponder on the individual vessels. Because of this complexity the raw data is prone to several errors. Usually AIS data is filtered to remove erroneous paths or ID conflicts. This paper utilizes the raw data, and the paths that cross land is interactively removed by filtering all the line-segments longer than a given tolerance distance.

In the current workflow of scientists and practitioners, the analysis of trajectory data is done by reducing the size of the problem, and/or aggregating it to a single pass-line, which then is statistically analyzed in greater detail. Such an approach gives a very good quantitative result, a single yes or no with respect to the considered hypothesis. This procedure, however, does not allow for a more flexible exploration of the data, aiding the forming of perhaps new and unexpected hypotheses that then are further analyzed.

In this paper, we demonstrate how a flexible visual analysis is utilized in this challenging application. We contribute a novel way of performing differential analysis of trajectory data and a new work-flow of iterating through these difference views. The presented solution was designed to quickly iterate through a sequence of difference views that are utilized to compare different categorical and quantitative attributes (such as different timespans, vessel-types, wind speeds), and to analyze a set of hypotheses as they emerge during the visual analysis. By using a visualization based on kernel density estimation to visualize the movement data, and difference views representing quantitative differences between the categories, the user can drill-down into the information. Possible correlations between waiting periods of vessels and bad weather conditions can be investigated, moreover, if the vessels' choice of a route is affected by the weather. Analyzing these and other questions supports the decision makers when evaluating whether or not to build the tunnel. In the next sections we first discuss related work, then we describe our application and the techniques employed here, then we analyze the domain questions, before we sum up, and provide a conclusion.

2 Related Work

A large number of publications deals with the visualization and analysis of time-dependent and multi-variate data (see Aigner et al. [2] and Fuchs and Hauser [45] for comprehensive surveys). Common analysis approaches for movement data include the visualization of raw data, computed summaries, or extracted patterns [3]. Spatial and/or temporal aggregation is often used in order to reduce the data complexity or visual cluttering. With such an approach, data items sharing the same spatiotemporal domain are summarized and depicted instead of the individual data values. According to Andrienko and Andrienko [4], data aggregation can be done either by calculating data characteristics (e.g., the sum, arithmetic mean, variance) or by grouping techniques such as clustering or binning.

BinX [14] visualizes long time series by binning along the time axis at different levels of aggregation and then displays mean, minimum, maximum value, and standard deviation per bin. Hao et al. [53] use pixel-based techniques to visualize time-dependent data at multiple resolutions based on importance values per data interval. Andrienko and Andrienko [5] visualize movement data as flow

maps where the spatial domain is subdivided into appropriate areas (based on significant points in the movement) and aggregated trajectories with common start and end points are visualized as arrows. Janoos et al. [63] analyze pedestrian movement data using a wavelet-based feature descriptor in order to detect anomalies. Grundy et al. [50] propose spherical scatterplots and histograms as an alternative representation of movement data.

A thorough overview on the usage of kernel and other density estimates in visualization is given by Scott [104]. Fisher visualizes the usage frequency of map tiles in his Hotmap [39], which is similar to a density estimate. Willems et al. [136] propose a visualization approach based on the convolution of dynamic movement data with a kernel, where the resulting density field is visualized as an illuminated height map. A combination of overview and details is provided by combining two fields, one computed with a small and one with a large kernel. While their approach provides very good results for presentation, it takes approximately 10 minutes to compute the data from one day (100.000 line segments). It is thus less suitable for a visual analysis where interaction is a key issue. Our approach, on the other hand, performs in real-time for even larger amounts of data using a GPU-based implementation. It is integrated in a framework of multiple views (with linking and brushing) and supports algebraic operations such as computing differences. Our approach also provides quantitative visualization where the value of a single pixel/cell shares the same unit as the depicted data.

Several applications support the visual analysis of temporal trends and patterns using interactive brushing or querying techniques. Interesting data subsets are interactively selected (brushed) directly on the screen, the relations are investigated in other linked views (compare to the XmdvTool [132]). Feature visualization and specification via brushing in multiple views (including histograms, scatterplots, and 3D views) is an integral part of the SimVis framework [31]. Jern and Franzén [65] propose a coordinated multiple views system for exploring spatio-temporal multivariate data. Hurter et al. [60] extract complex features in aircraft trajectories by brushing in juxtaposed views. The brushed trajectories are spread across views with a pick and drop operation. The views can be rapidly configured by connecting data attributes to visual variables such as color or size. Kehrer et al. [70] recently demonstrate how the iterative reconfiguration of depicted view attributes enable a powerful analysis process. In our system, we explicitly represent such transformations of views that support the visual analysis of a set of hypotheses that emerge during the visual analysis (e.g., comparing traffic at different workdays).

According to Verma and Pang [126], different data sets can be compared at the image level, the data level, or the feature level. Image-level comparisons include side-by-side visualization and the visualization of differences between the images per pixel. For such approaches, also the selection of an appropriate color map is very important (e.g., using a diverging map to visualize differences [19]). Polaris/Tableau [112] supports the visual exploration of hierar-

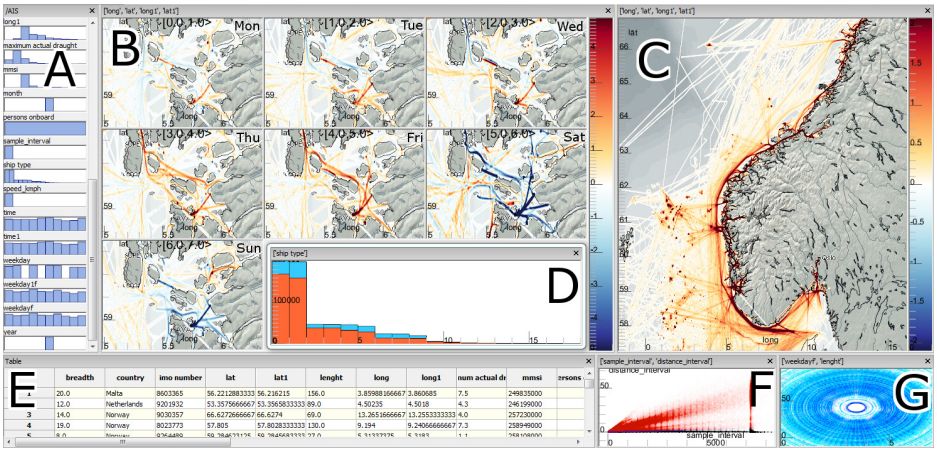


Figure 2: Overview of the described application. A shows the available attributes from the dataset, displayed as histograms. The 7 multiples in B show a close-up to Stavanger, split by weekdays, showing the differences, in traffic volume, from the average. C is an overview and D is a histogram of different ship types. E is a table view of all samples, F a scatterplot, and G a radial plot displaying vessel activity during the week.

chically organized multi-variate data using a table-based layout of views (commonly called small multiples [121]). For side-by-side comparison, user-specified categories/hierarchies are opposed such as time (year, quarter, month), products, or spatial locations (town, state, country). Data attributes can, moreover, be interactively transformed (e.g., by aggregation or grouping), filtered, and/or brushed. Woodering and Shen [141] propose volume shaders to compare and combine multiple time-dependent volumes by consecutive algebraic set operators and numerical operators. For interaction and visualization of the resulting volume tree they utilize image spreadsheets (compare also to Jankun-Kelly and Ma [62]).

3 Interactive Difference Views

The interactive visual analysis and exploration of the movement data is carried out in a setup of coordinated multiple views with linking and brushing (see Figure 2). The views include histograms, scatterplots, and frequency-based views based on Kernel Density Estimates [108] (KDE). The latter views are computed by convolving the movement data with a kernel (usually a Gaussian) for each sample, resulting in a density estimate that can also be extended to cope with trajectories (using a line kernel instead of a point spread function). The questions of our application partners were answered by developing an iterative workflow for creating quantitative difference views, with the aim of facilitating the fast

and flexible investigation of large amounts of movement data. A difference view results from subtracting one KDE plot from another one, which then shows the quantitative difference between them. While animations and side-by-side views often provide good means to answer qualitative questions (e.g., "where" and "when"), they are less suitable for answering quantitative questions (e.g., "how much/many"). Such quantitative differences are explicitly represented in our difference views, for instance, using a diverging color-map [19] (see Figure 2 B).

In the following, we describe our interactive and iterative analysis, our quantitative difference visualizations, and how we can handle large datasets at interactive frame-rates.

3.1 Interactive and Iterative Visual Analysis

From the domain questions, we derived a couple of requirements for our solution. As often in the context of hypothesis testing and analysis, every new finding leads to new questions as well. Accordingly, we shaped our application in an as iterative and interactive fashion as possible. This iterative workflow enables the user to search for one answer, and then further investigate unexpected trends, or to search for multiple indicators forming a single answer.

When the user first loads a multivariate dataset, an overview of the attributes (variates) is automatically displayed in the dataset window (see Figure 2 A). Every attribute is represented with its own small histogram. These histograms acts as drag-sources, in a drag and drop sense. To construct a visualization, the user drags an attribute onto an empty view. While still dragging, a context frame appears over the current view, with multiple possible drop targets. Each of these drop targets represents a possible binding between the dragged attribute and a property of the current visualization, e.g., a spatial binding to either the x or the y axis of the view, or binding to size or color. In this manner the user quickly creates one or several compound views.

The next step is to relate these views by brushing, and specify features across multiple variates by constructing a set of rules. As an example, the user brushes all northbound vessels with a speed of 5 knots or more in one view, and selects the category of ship-type equal to Tankers in another view. This ruleset is then reflected on all views, using a focus+context style in sample-based views (e.g., scatterplots), and filtering in the KDE plots. As another step on top of these relate-and-filter techniques, we have added a *compare over* expansion. When the user drags an attribute to a frequency-based view, he or she has the option of dropping this to the compare-over option (available from a context menu). This expansion splits the current view into one difference view for each of the categories (or bins if the attribute is continuous). Each of these difference views then displays all the samples matching the given category subtracted by the average. In areas where this category is greater than the average, the result will have a positive sign (red in our figures), and negative (blue) in areas where there

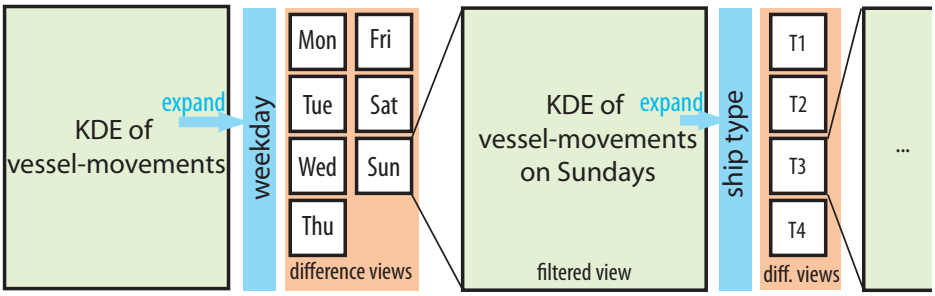


Figure 3: Iterative data exploration via difference views.

is less than average. Further below, we discuss in more detail what we mean by computing the average with respect to categorical attributes such as weekdays, ship-type, or wind direction. These difference views build on top of the existing ruleset from the previous step, and thus form a two-level rule hierarchy. So as in the above mentioned example, when expanding vessel traffic over weekdays in a map plot, this would show how the northbound tankers with a speed of 5 knots or more, on one weekday, would compare to the average weekday of northbound tankers.

After creating several difference views, the user can select one particular difference view. This view then replaces the previous reference view, and its second-level rule on a category will be added to the level-one ruleset. In Figure 3 we describe this iterative creation of difference views, where categories are selected through a series of difference views. Returning to our previous example, where we had seven difference views, one for each weekday (category), we can select one day, e.g., Sunday, and then all views only show northbound tankers, with 5 knots or more on Sundays. This cycle can be repeated, enabling a deep drill-down into the data.

3.2 Quantitative Difference Visualizations

The concept of difference views, and their ability to display quantitative differences between two comparable views, has been utilized in several other works, yet there is no clear workflow for the flexible configuration of exactly what to create difference views between. To facilitate the creation of meaningful difference views, we defined the *compare over* functionality, which splits up a current view into several, one for each category. For example, the top view of Figure 4 shows a frequency view of horsepower vs. miles per gallon of the 406 cars in the Car dataset [95]. After this view has been customized or optionally filtered, the user then drags the origin column (denoting which continent the cars are produced in) into this view, and then drops it on the expand icon in the in-screen menu

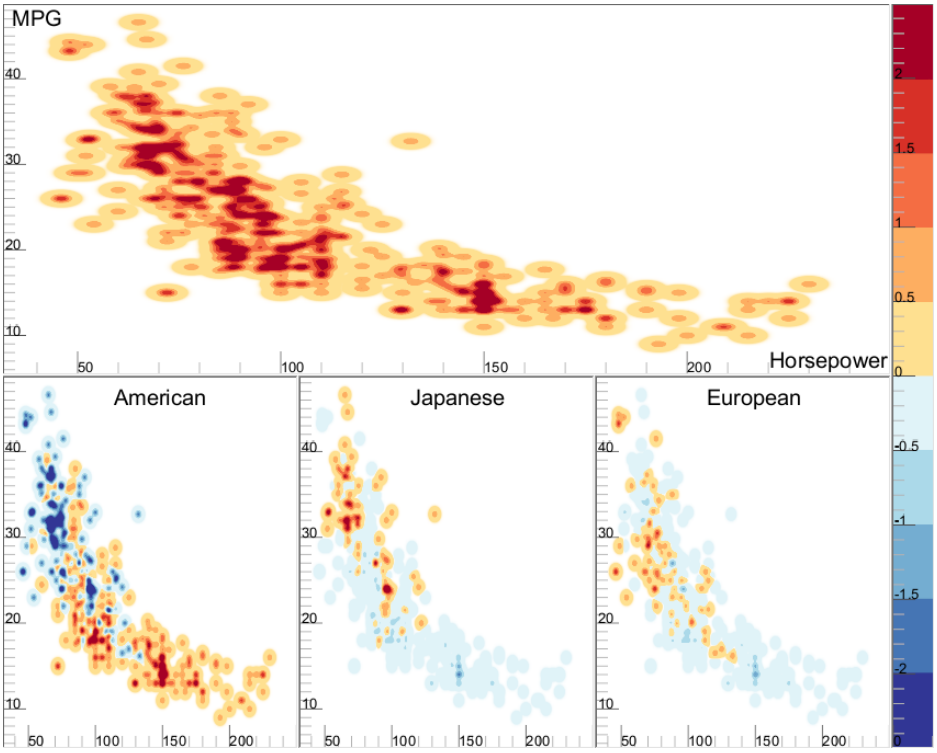


Figure 4: Miles per gallon (MPG) over horsepower for 406 cars [95] shows an inverse correlation in the top view. The top view can be expanded in the three bottom views, where American, Japanese, and European cars are compared to the average. We can see that American cars have many more cars with high horsepower. Compared to European cars, they have more horsepower for an equally rated MPG.

that pops up. The whole of Figure 4 is the automatic result of this operation. Since the column dragged is a categorical attribute the user is presented with one additional view per category. These views present how samples in this category compare to the average over all categories. The average in this case is achieved by dividing by the number of categories. The compared difference views show the sum of each sample's kernel, where those samples from the current category are given a positive sign, all others a negative sign, and all scaled for averaging. A 2D KDE [104] is defined by

$$\widehat{f}_{\mathbf{H}}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_{\mathbf{H}}(\mathbf{x} - \mathbf{x}_i)$$

with $\mathbf{H}^{-\frac{1}{2}}$ being a symmetric and positive definite bandwidth matrix and $K_{\mathbf{H}}$ being defined as

$$K_{\mathbf{H}}(\mathbf{x}) = |\mathbf{H}|^{-\frac{1}{2}} \mathcal{K}(\mathbf{H}^{-\frac{1}{2}}\mathbf{x}).$$

$\mathcal{K}$ is a multi-variate kernel function that integrates to 1. By defining result of two KDEs, $f(\mathbf{x})$ as the average view, which includes all samples, and $g(\mathbf{x})$ for the subset of only those samples within the given category, we can define our difference view as:

$$d(\mathbf{x}) = g(\mathbf{x}) - f(\mathbf{x}) \tag{1}$$

Instead of first creating the full KDE $f(\mathbf{x})$, and then subtract a subset of samples from that KDE, we can do this in a single step. Since the set of those samples matching the category is a subset of all samples, we can simplify Eq. 1 to a single summation pass over the samples, and scale the samples in the category by $\frac{1-n}{n}$, and the rest by $\frac{1}{n}$.

As an another example, considering temporal ranges, we look at how traffic differs on the different days of the month. To establish the KDE representing an average day, we calculate the temporal range of all the samples, i.e., the number of days our sampleset spans, and divide by this range. Next the temporal range of the samples in the current category needs to be calculated, which is not quite as trivial as the example above. If we have a set of samples spanning over several months, and would like to compare weekdays against weekends. Since there are more days belonging to the weekday category than that of the weekend category, we cannot normalize by the total number of days. Instead, we need to count the number of days matching "weekday" that actually contributes to this subset.

In our solution we have implemented an automatic technique that iterates over the samples and can calculate the sum of smaller temporal ranges that match the current category, e.g., days, hours with particularly strong wind, or weekdays. When this is established, the temporal difference view can also be calculated in a single step using the above equation.

3.3 Large Datasets

A requirement on the application was to enable the analysis of statistical significance of the results. Significance, here, is determined by the amount of noise, the signal and sample size. Since we do not have any influence on the contained noise and the signal size (after data acquisition), we attempt to optimize the quantitative significance of our analysis through the third factor, i.e., the sample size. Allowing for larger datasets to be interactively analyzed, helps to increase the confidence in the extracted findings. If we had choosen to support a sample size just large enough for the task at hand, this would not allow for any flexibility with respect to further drill-down steps or alternative comparisons.

In visualization we often deal with three levels of limitations on dataset size, (1) when the dataset fits into graphics memory, (2) when it is too large for

graphics memory, but still fits in main memory; and (3), when it is too large to fit in main memory, but reside on a file level. Our implementation supports the third category, but in order to keep interactivity, employs a three level data handling scheme. If a file is larger than what the application can hold in main memory, only a subset of the file is loaded. A yet smaller subset is then kept on the graphics card, and displayed at all times in the application. The size of this smallest subset is selected such that interactive speeds can give quick response when brushing, even though there are many views. Immediately when interaction ceases, the application starts rendering, in batches, to the now stationary views from the rest of data in main memory. And again, when interaction starts again, all views fall back to only display the GPU-resident data in the first place. This interaction is shown in the supplementary material. These two top levels gives quick and interactive access to what should be a representative sub-sampled portion of the data. Due to the nature of interactive visual analysis, including filtering and refining, we do not stop there. We allow the application to keep a second file of query results, that allows the user to apply the visual brushes to the file level. This new query results file can then be used for further analysis, and then perhaps have all its data fit in main memory, and thus have it all shown in the visualization views.

4 Answering the Application Questions

The main question that the government wants answered is whether or not to build a tunnel through Stad, and such an answer should include reasons as to why, backed up by quantitative indicators. In this analysis, the domain expert investigates potential decision criteria, and then investigates whether those are significant or not. In this work, the domain expert contacted us with an interest in visualizing the AIS data, and to look at two such indicators. First, it was interesting to look at how many vessels are actually waiting when there is bad weather, and second, whether vessels go closer to the shore when the wind picks up, which increases the risk of accidents. To compare our AIS data with weather data, we obtained meteorological measurements from two stations, Kråkenes fyr, a prominent light house south of Stad, and Svinøy fyr, another light house on a small island north of Stad. These measurements contain wind direction and wind speed, which we then applied to all samples based on their spatio-temporal proximity.

To compare how wind speed affects the amount of stationary vessels, we brush vessels with speed close to zero. We then zoom into the area around Stad on the map view, which now shows areas where vessels are stationary. To compare how this view changes with respect to different wind speeds, we take the wind speed attribute and perform a compare over action. The result is shown in Figure 5. The top-left category (weak winds) shows a greater than average amount of

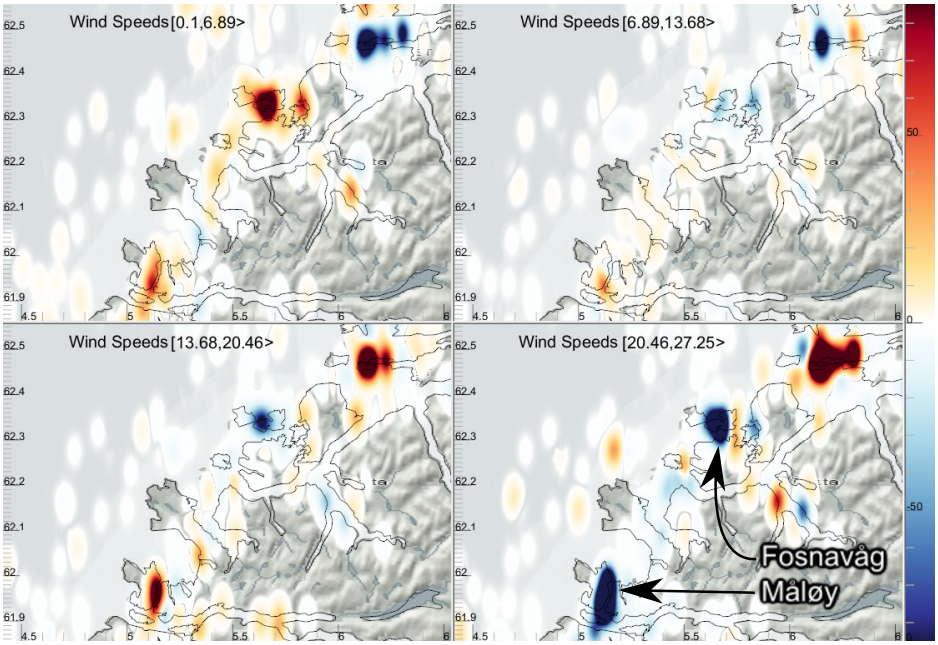


Figure 5: Close to Stad, brushed to only include stationary vessels. The views show how many (compared to all) vessels are stationary, given different wind speeds.

stationary vessels. The bottom-right view with the strongest winds (strong gale and worse) shows a significant drop (7%). The top right view is 3.2% below average and the bottom left 5% above average. Accordingly, there is no trend that either confirms nor reject our hypothesis yet. The explanation for the increase in stationary vessels when the weather is good, is that there is an overall more vessels out at sea when the opportunity calls for it, and the opposite for the strongest winds. To show this, we then include all vessels, stationary or not, but keep our wind categories, and calculate the integrals. This reveals that there is a drop in overall traffic of 20% in the strongest wind category, and an increase in traffic in the lowest category. An overall drop in 20% traffic, but only a drop in 5% stationary vessels, indicates that our hypothesis holds, and that there is indeed an increasing amount of stationary vessels when the wind is bad.

Another approach is to compare the actual traffic past Stad, and to compare how this volume is changing with respect to different weather conditions. If our earlier assumption that vessels need to wait when there is bad weather is true, we should see a decrease in traffic. Figure 6 shows the traffic around Stad expanded into four categories of wind speed. By computing the integral for a selection, we can see that the first category (no/little wind), shows 8.6% more traffic than

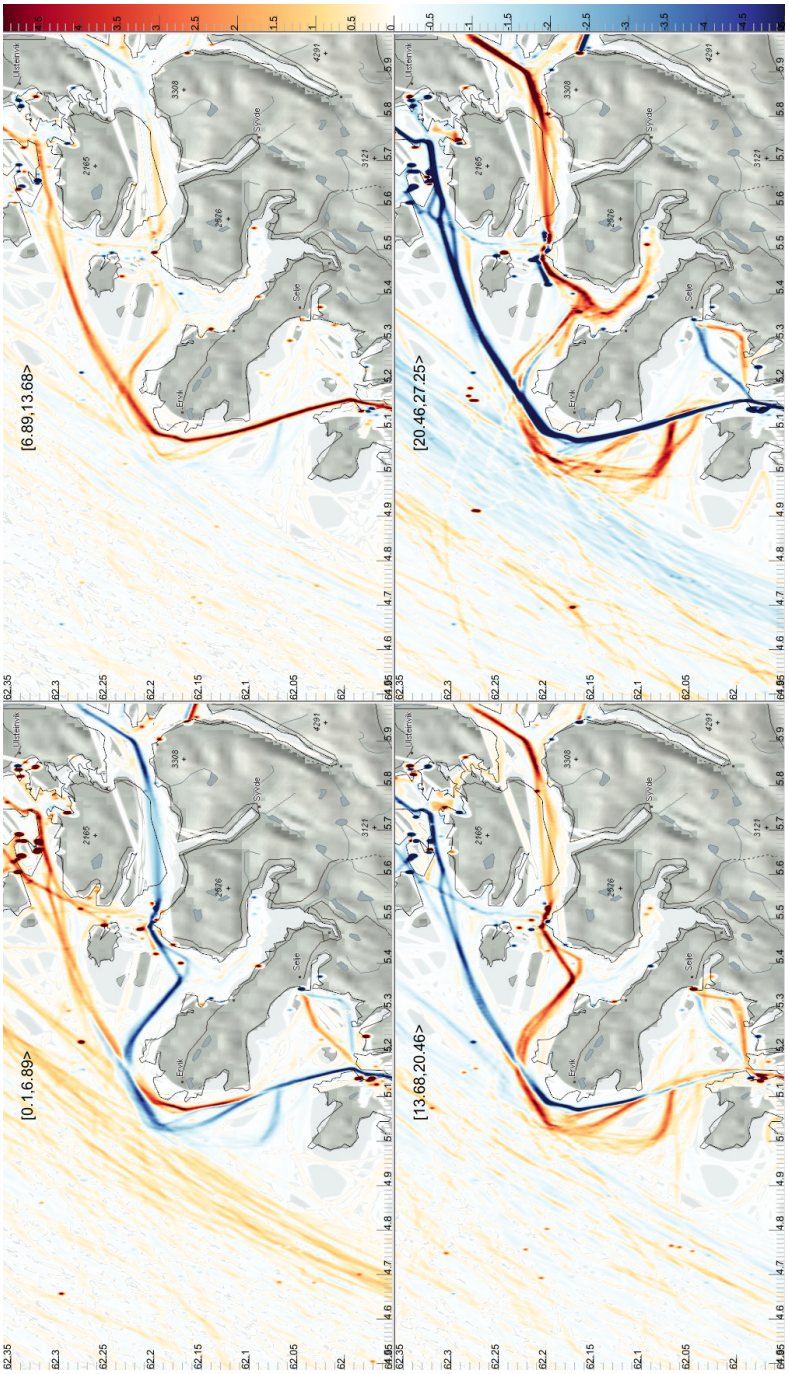


Figure 6: Passing vessels outside the Stad peninsula, and their changed movement pattern given stronger winds. Red colors indicate more than average traffic in that interval of winds, and blue colors indicate less than average.

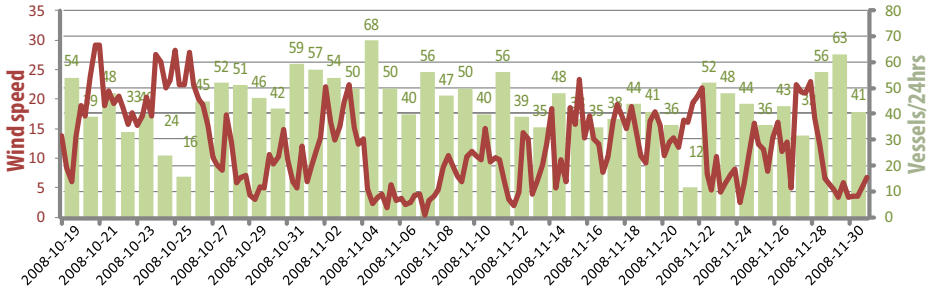


Figure 7: Wind speed in m/s and passes by Stad, peaks in wind speed forces vessels to wait, and then when the weather gets better, there is a increase of vessels passages.

the average, the next 3.0% more than the average, the third 5.6% less and the last, with winds from strong gale and up, show a significant decrease by 24%. A finding that further strengthens our inital hypothesis, however, we can even investigate further.

The third approach is a more item-based one; we can read from the weather data that 21st of November 2008 had particularly bad weather, and that the following day the weather calmed to a breeze. By counting the number of vessels that passed Stad on these two days, we can see if together they stay within the average passes, and how many have been delayed by one day. By brushing an area around Stad and one single day, the table view (see Figure 2 E), will display that the selected area has an average of 38 unique vessel IDs registered per day, on the 21st there were 12 and on the next day 47. Figure 7 show this as well, where the day following the storm had higher traffic. Using these averages, over more than just this case, we could calculate on average how many vessel hours are lost during a season. Comparing to the average in just this one case, however, one can estimate that around ten vessels would have a delay of 24 hours, or, 240 hours lost on a storm lasting less than a day.

The other investigated indicator is whether vessels draws closer to shore when the weather gets bad. In the previous paragraph, we discussed Figure 6 which computed the integral of a selection to see quantitative differences. Answering this question of vessel paths can also be done by studying the same figure. In the first of these four figures, the one with the lowest wind speeds (top-left) there is a red curve going close to shore, which means that there is a greater than average amount of vessels taking this route when the weather is good. Additionally, in this same figure, the clearly defined blue route further out, defines that there is a much less than average amounts of vessels taking this route. In the next wind category, top-right, this outer route is now “invisible”, which means that there is an exact average amount of vessels taking this route. The route close to shore still contains a greater than average amount of vessels. In the third wind

category the route close to shore is now clearly defined blue, and those who pass Stad does so selecting the route further away from shore. Similarly with the fourth category, with winds of strong gale or stronger, the route close to shore contains close to zero vessels. Moreover in the two last categories more vessels go straight towards the safety of inshore, where in the two first categories, vessels take the more exposed "shortcut" straight over to Herøy (the island in the top right corner of this map). So in conclusion, Figure 6 clearly shows an opposite effect than the original question, meaning that the stronger the winds, the more distant routes from the shore are selected.

In our discussion with domain scientists, they stated that our application gave them an improved insight into the complexity of their original questions; an insight that later also strengthened their value on AIS as an asset for analysis, which was not fully realized before. Our use of AIS as a probability density estimate, enabled both a non-parametric exploration of the entire dataset, and an in depth analysis of selected details. Furthermore, they found the interactive analysis of AIS as a frequency view “groundbreaking”. The application was both flexible and understandable for the users, and showed a great potential for further analysis. Previous analysis required extensive manual labor, and provided statistical analysis for a few chosen pass-lines; this application would alleviate this labor, by providing similar details for every pixel/cell, with a simplified analysis work-flow.

5 Summary and Conclusions

In this paper, we presented an application to investigate particular questions presented by the Norwegian Coastal Administration (NCA). NCA will use conclusive answers to these questions as indicators in their recommendation to the Norwegian government, as to whether or not build a tunnel through Stad. On the first question, concerning the correlation of waiting periods and bad weather conditions, we showed that even with a total reduction, by 24%, in the traffic when there is strong winds; the proportion of the traffic that is stationary vs. the traffic passing Stad is increasing with increasing wind speeds. Another conclusion on this case is found by a sample-based approach, which shows that there is a temporary increase of passings by Stad, after periods of strong winds. On the next question, on whether bad weather affects the vessels to choose a route closer to shore, Figure 6 shows an opposite effect, meaning that more distant routes from the shore are chosen when there are stronger winds. We have demonstrated how this application, using the techniques of iterative creation of difference views and through the use of quantitative visualizations, reached conclusions to the questions posed, and the flexibility to search for several alternative indicators, and thus also meet future demands.

6 Acknowledgements

The work presented here is a part of the project "e-Centre Laboratory for Automated Drilling Processes" (eLAD), participated by International Research Institute of Stavanger, Christian Michelsen Research and Institute for Energy Technology. The eLAD project is funded by grants from the Research Council of Norway (Petromaks Project 176018/S30, 2007-2010), Statoil ASA and ConocoPhillips Norway.

Paper E

Interactive Model Prototyping in Visualization Space

Ove Daae Lampe[1,2], and Helwig Hauser[1]

[1]Department of Informatics, University of Bergen, Norway
[2]Christian Michelsen Research, Norway

Abstract

Researching formal models that explain selected natural phenomena of interest is a central aspect of most scientific work. A tested and confirmed model can be the key to classification, knowledge crystallization, and prediction. With this paper we propose a new approach to rapidly draft, fit and quantify model prototypes in visualization space. We also show that these models can provide important insights and accurate metrics about the original data. Using our technique, which is similar to the statistical concept of de-trending, data that behaves according to the model is de-emphasized, leaving only outliers and potential model flaws for further inspection. Moreover, we provide several techniques to assist the user in the process of prototyping such models. We demonstrate the usability of this approach in the context of the analysis of streaming process data from the Norwegian oil and gas industry, and on weather data, investigating the distribution of temperatures over the course of a year.

This article is submitted to SIGRAD 2011 in Stockholm, Sweden.

1 Introduction

Modeling is an essential part of scientific work. To be able to learn from observations and to utilize the gained knowledge for subsequent analysis, such as prediction, the modeling of the observed phenomenon in some sort of a prototype is crucial. Also, central to science is that model hypotheses are tested, refined and validated, or possibly rejected after testing. In the following, we consider a *model* to be a physical, mathematical, or logical representation of a system entity, a natural phenomenon or process, and that *modeling* is the act of creating a model [109]. In experiments or, as we will focus on, modern process logging, measurements and data is gathered. Establishing a model on measured data often start by employing empirical / statistical tests with trial and error, finally ending up with a *model prototype* and the statistical confidence on the model's accuracy. When and if the model prototypes hold up to scrutiny, one can aim to generalize these, and create a *model template*. The model template can be thought of as a scale invariant model, something that would fit to data irrespective to influencing factors, and then used to quantify these factors. Eg. a model template of time over height squared would, if applied to Galileo's raw experimental data, establish the gravity constant, along with the statistical confidence of this value.

Considering the visualization of particle paths in a tokamak (fusion reactor) as another example, we first consider that the most obvious footprint of direct data visualization is the fact that the particles intensely rotate – an observed phenomenon that is principally important, but not really surprising. To see it in a visualization is interesting and useful for a moment, but not really for much longer. Only shortly after confirming the expected rotation of particles, we want to proceed and look behind this phenomenom: is there any secondary motion characteristic to be seen? To actually check such a hypothesis, we can aim at abstracting already understood and accepted aspects of the investigated phenomenon from the data visualization. This abstraction leads to three results: (a) the finding itself, which will undergo an externalization, where the finding is pulled out of the visualization represented in a different form, and (b) a residual data visualization – where the finding has been subtracted from – which then allows for studying remaining aspects of the observed phenomenon that do not follow the model. In the case of the tokamak example, we can think of an abstracted visualization of the particles, e.g., by using a Poincaré map (the main feature, i.e., the rotation of the particles, is then no longer visible, but only off-rotation deviations of the particle paths). This clears the view and allows the user to gain a more thorough understanding of complex phenomena through iterated analysis, including modeling and abstraction. (c) since large scale movements or densities of data is subtracted after the abstraction, this enables the further study of features which might be a magnitude smaller than the overshadowing and perhaps obvious features.

With this paper we aim to introduce a novel iterative workflow of assisted modelling, abstraction and subtraction to completely map a dataset from the originally visualized view to an abstracted and quantified one. To achieve this goal, we first provide a novel technique to assist a user to sketch locally optimal models, and then how to represent these in an abstract and quantified manner.

The remainder of this paper is organized as follows: Next we discuss some related work. Then we elaborate on the theoretical part of this work in section 3, before we go into detail with respect to our technique in section 4. In section 5 we present results from the application of our approach and demonstrate its usefulness in this context.

2 Related Work

Extracting well defined features is a related topic often studied in the field of flow visualization. Post et al. [94] provides a good overview of the current state of art and how the features are found, abstracted, and quantified. On other phenomenons where the basic models is understood, data for visualization can often be reduced or abstracted. Löffelmann et al. [82] use Poincaré maps as such a technique to create abstractions of data, reducing the dimensionality, and thus allowing the visualization of secondary features. On data in which the model is not understood, Rheingans and desJardins showed that inductive learning techniques, such as self organizing maps (SOM), can construct explanatory models for large, high-dimensional data sets [30, 100]. Their technique employs an overlay of models and data visualization, and thus creates an implicit visual comparison of model vs. data. Models as such are used in all sorts of scientific work – it is therefore perhaps beyond the scope of this work to reasonably discuss the role of models in science and visualization. Examples reach as far as into model-based segmentation of medical data (for example research for cardiac diagnosis [144]) or into model-based object recognition (such as for robot vision [24], for example). These approaches show how models are used for classification and segmentation. Often, they also utilize a difference view, in which they show the match of a model to the data (which here relates to our residual data visualization). By iteratively adopting our technique, we can find higher order features, which are related to the field of multi-resolution analysis and multi-scale modeling, such as wavelet-based approaches [129], for example. However, instead of focusing on a decomposition in frequency domain, we capitalize on partial abstraction which is feature-based and local. For a more principle/philosophical discussion of models in science, we refer to the corresponding entry in the Stanford Encyclopedia of Philosophy [44].

The obvious step after extracting features is to put them to good use. Liu and Stasko [80] investigate how internal representations (mental models) and external visualizations relate to each other. The authors state that such mental models

are used during visual reasoning to "simulate" the behavior of the corresponding visualization system [80]. Our approach helps the analyst to externalize such mental models, and compare the data to it. Shrinivasan and V. Wijk and Yang et al. have investigated how to effectively support an *externalization* of findings in visualization. Yang et al. [142] describe a system which allows users to externalize findings, or *nuggets*, while exploring a dataset. These nuggets are then added to a *Nugget Management System*, where clustering and meta-information, help the sense making process. They also describe how visualization in this nugget space prove useful as an abstraction of the original data. Shrinivasan and van Wijk present the *Knowledge View* [107] in which not only the findings are externalized, but also the interaction path that lead to it. Their user study shows favorable results with respect to externalizing knowledge in mind maps. Wohlfart and Hauser used the concept of storytelling [138] as a way of externalizing and communicating findings. This work allow an analyst to create a visual story, which then can be played back to another user, with varying degree of interactivity.

The visualization scheme utilized in this work, is highly dependent on a frequency based technique that also support meaningfull difference views. Daae Lampe and Hauser presented a technique [28] that enables the continuous distribution of data, using kernel density estimates (KDE). This technique also extends to support the continuous distribution of data-samples that is temporally connected, in that it creates a line-kernel that connects these samples. Daae Lampe et al. also effectively utilized these 2D KDE techniques for difference views [29] in an application that aimed to generalize how to create multiple views that highlight the differences in distributions between distinct categories.

3 The Basic Idea

One of the goals of this research is to effectively support practitioners and scientists to analyze process data that is streaming in or updated at considerable rates. We provide an approach that allows users to rapidly prototype models for structures which the user identifies in a visualization of the data. These model prototypes act (1) as parts of the externalization of the user findings and (2) as means to quantify the structures for subsequent user tasks. Accordingly, we first focus on how to identify structures which lend themselves to model prototyping. We see two opportunities: Either the user has a conceptual model of what to look for (analytic/confirmative setting), or she/he aims at creating one by looking at the data (explorative setting). In the first case, it is useful to integrate the anticipated model within the visualization, to get initial information on how well the data fits the model. In the second case, it is useful to have a visualization that supports the user in interactively prototyping the model to then subtract it from the visualization, and get immediate feedback on how well it fits. As an example of these settings we use the case in Fig. 2, where a normal distribution is

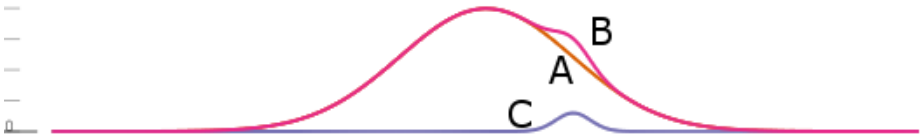


Figure 1: $A(x) = N(0,1), C(x) = 0.05 * N(1,0.2)$ and $B(x) = A(x) + C(x)$

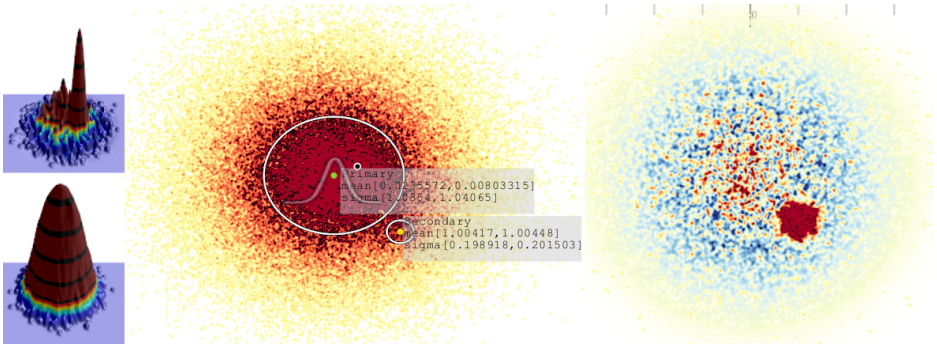


Figure 2: This is the 2D version of the synthetic example from Fig. 1. The left images shows the (logarithmic) height-map after and before subtraction. The middle image shows the quantified measures as read out from both the primary and the secondary feature (after fitting two model prototypes). The image on the right shows the data, after abstracting the primary feature, clearly revealing the secondary feature, even though it was almost completely hidden.

occluded by another. In the analytical setting, the user has a hypothesis that the data represents a normal distribution with a specific mean and standard deviation, and create a prototype with these settings to confirm or reject this. In the explorative setting, the user would look at the visualization and by observation suspect that this is a normal distribution. The user would then pick the normal distribution model template, and apply it to the visualization, to create an initial model prototype. By interactive sketching and fitting, the user will either reject this hypothesis, or as in the case of Fig 2 get it confirmed. By the externalization of this prototype, the user would also get accurate measures of mean and deviation; in addition to reveal the previously occluded secondary feature.

In this interactive externalization we follow the workflow: *visualize and observe, sketch and fit, externalize and subtract, then iterate*, as shown in Fig. 3. This figure is read from top left then right or down. The data is visualized, and by observation an interesting feature is detected. The user selects a suitable model template and by sketching onto the visualization, creates an initial model prototype. Through further sketching and automatic fitting, the prototype is finalized. This complete prototype is externalized to model space, and a residual

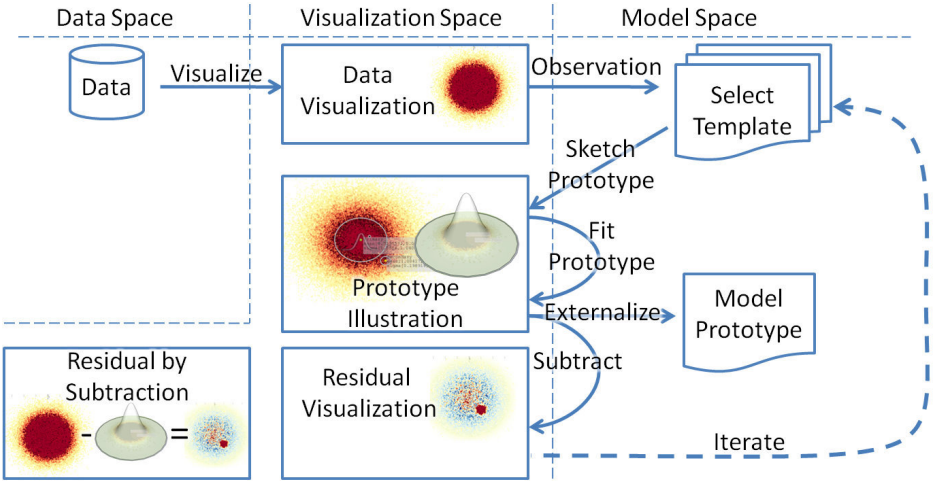


Figure 3: Our proposed workflow: visualize and observe, sketch and fit, externalize and subtract, then iterate.

visualization is created by subtracting the model prototype from the data visualization. At this point the procedure can be repeated by observing another feature in the residual visualization, selecting another template to prototype, and so on.

To illustrate this idea, again consider the dataset as shown in Fig. 2 as a full day's operation of a hypothetical process. This operation went as planned, with the exception of one hour where several values where abnormal, represented by the secondary feature. The abnormal values are initially completely occluded by the normal operation. By introducing a model for "normal operation", and then subtract it from our visualization, we are left with that which does not fit normal operation. Now it is up to the user to further investigate the features that stand out, i.e. the hour with abnormal values, to prototype and perhaps further subtract this data as well, given existing explanations. Generally we can assume that this sequence of visualize and observe, sketch and fit, externalize and subtract, is iterated for as many relevant features as show up. Visualizations which have all its features modeled, contain only data which behaves according to understood models. At that point the externalized model prototypes serve another purpose, which is a much more condensed form of information, than a series of data-samples, namely quantitative parameters.

To support this workflow we separate our visualization into three major parts, the data visualization, the model prototype illustration, and the residual visualization. The model prototype illustration serves the purpose of giving a non-occluding and condensed view of where all the previous and the current model prototypes are located. Additionally, the prototype illustrations act as handles for interaction. The residual visualization serves several purposes. The first pur-

pose is to show how well the model fits the data, and the second one is to then utilize the *visual range* better (through a scale-up operation). Human perception, and thus visualization, has a limited tolerance for range, e.g., there is finite limit to how many colors we can distinguish, or a limited range in how we can perceive brightness. By subtracting low frequency, high amplitude, features, we can effectively and automatically create a new and optimized range.

3.1 Visualization

Streaming process data requires a direct in situ visualization of the data, since it is constantly updated. The visualization technique utilized here, is based on work by Daae Lampe and Hauser. [28], and 2D Kernel Density Estimates (KDE). This technique visualizes a large set of samples, but displaying the convolved sum of kernels, one per sample, resulting in an analytical density function, that supports meaningfull difference views [29]. Additionally, the usage of scaled kernels will create a visualization that shows the distribution of time, independent of sampling rate.

3.2 Model Sketching and Fitting

In computer science terms, a model template would be a class, and a model prototype would be an instance of such a class. In the process of creating a prototype, *sketching* is considered the manual input, and *fitting* the automatic algorithm assisting the user. Model templates come with properties, that the prototype needs to instantiate, which we categorize below. We consider them to be *shape*, *distribution*, and *scale*.

Shape characterizes the form of the model *along* the sequence of samples (after visualization). A linear structure can be described by a line model, more complex forms would follow spline curves, for example. In our case, we are fine with a piecewise linear model template. However, more complex models are equally possible (as long as a fitting procedure, see below, is available, as well). We refer to this central characteristic of a model as the *shape construct*. Selecting a shape requires the selection of the following parameters, *shape construct* and *control points*. We will only consider the following subset of shapes for the remainder of this paper.

- Single Point Construct, which will fit data with no linear correlation (see Fig. 2).
- Piecewise Linear Construct, which will fit data with a correlation (but not necesarrily a linear correlation, see Fig. 8).

Distribution determines the form of the model *across* the sequence of samples. Whether it is due to noise, weak measurements, or other natural phenomena, real

world data rarely ever line up perfectly. We therefore consider a certain data distribution across the sequence of data samples, which we model accordingly. The definition of the distribution we will refer to as the *distribution construct*. Selecting a distribution requires the selection of the following parameters, *distribution construct* and *width*. In the following we will denote this width as $\mathbf{r}$, a vector separating the "radius" in the screen space coordinates u and v.

Scale, finally, is a measure of intensity. Depending on the visualization parameters, this parameter will have different meanings. E.g. for a box, the scale will be the average within it, for other, it will give a more complex measure of the intensity within the model.

Summing this up, we need to find a matching shape construct, a matching position, select a distribution construct, find the distribution width, and finally determine the scale, when we aim at fitting the model prototype to the data. To measure how well a model prototype fits the data, a problem not very different from image comparison, a correlation function like sum of squared differences has proven to be useful [47]. Other difference norms, such as the L1 norm, for example, also are possible and the choice of which norm to use is usually application-dependent. After choosing a squared differences norm (here L2), we investigate the opportunities to minimize it for fitting. To simplify the function to minimize we will consider the selection of shape construct and of distribution construct as selected manually by the user (according to his or her a priori assumptions about the data). In the following, we denote the discrete scalar field, which results from mapping the data to visualization space a $D(u, v)$, where u and v are the screen or canvas coordinates, and the models scalar field (also after mapping into visualization space) as $M(u, v)$. To generate this scalar field M, we first need to select the shape position $\mathbf{p}$, the distributions radius/extension $\mathbf{r}$, and the scale/height h. Based on these selections we have a function $L(\mathbf{p}, \mathbf{r}, h)$ which when mapped to the visualization space represents M

$$L(\mathbf{p}, \mathbf{r}, h) \rightarrow M(u, v) \tag{1}$$

Defining $\mathbb{UV}$ as the natural numbers from zero to canvas-width and canvas-height we get the difference measure (L2):

$$s = \sum_{(u,v)\in\mathbb{UV}} \left(D(u, v) - M(u, v)\right)^2 \tag{2}$$

From eq. (1) and (2) we find that s, the sum of squared differences, is a function $s(\mathbf{p}, \mathbf{r}, h)$. From this we can extract our target variables through the following optimization:

$$\underset{\mathbf{p}\in\mathbb{UV},\ \mathbf{r}\in\mathbb{R}>0,\ h\in\mathbb{R}>0}{\operatorname{argmin}} \quad s(\mathbf{p}, \mathbf{r}, h) \tag{3}$$

Optimizing this equation is not straight forward, but since the user inherently sketches close to the desired solution, we can avoid potential problems with locating the global optimum, and only aim for the local minima. A restriction we

introduce, to prevent the shape constructs position from degenerating, is to only allow position corrections orthogonally to the direction of this points line segment. If the construct only has one point, we allow movements in all directions during fitting.

We experienced satisfying results with the traditional Newton's method for this optimization problem, due to its good convergence [88] in local problems. Newton's formula:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

finds roots of $f(x)$ on the basis of $f'(x)$. By using the Taylor expansion on $f(x)$, we find that the sequence x_n defined as

$$x_{n+1} = x_n - \frac{f'(x_n)}{f''(x_n)},\ n \geq 0 \tag{4}$$

which can be generalized to several dimensions by replacing the derivative with the gradient, $\nabla f(\mathbf{x})$, and the reciprocal of the second derivative with the inverse of the Hessian matrix, $Hf(\mathbf{x})$ leading to the following iterative scheme:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [Hf(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n),\ n \geq 0. \tag{5}$$

Eq. 5 will converge towards our desired solution, given that $f(x)$ is twice-differentiable and our initial estimate x_0 is close to the solution. Calculating the Hessian matrix (or inverse thereof) is an computationally expensive operation, and we separate the derivatives and to minimize the function s by individually minimizing the variables in order of their influence of the overall model field

$$\underset{\mathbf{p}\in\mathbb{U}\mathbb{V}}{\operatorname{argmin}}\, s(\mathbf{p}), \quad \underset{\mathbf{r}\in\mathbb{R}>0}{\operatorname{argmin}}\, s(\mathbf{r}), \quad \underset{h\in\mathbb{R}>0}{\operatorname{argmin}}\, s(h) \tag{6}$$

We look into these optimizations in section 4 with measured results of convergence.

3.3 Quantification and Model Prototyping

After completing a number of model prototypes in visualization space, we transfer quantitative information back from visualization space into *model space*, a technique called externalization. Model space can be thought of as a summary of the understood features found in the data, and thus a more holistic approach to modeling is possible; one that also takes model parameters into account, which haven't dealt with in visualization space. For example, when visualizing speed vs. height of an object in free fall, this only can lead to model prototypes correlating those two parameters, not (yet) considering other potentially influencing factors, such as aerodynamic drag, etc. As established in section 3.2, the information available is the position $\mathbf{p}$, distribution extent/radius $\mathbf{r}$, and scale h. Transferring $\mathbf{p}$ to model-space is trivial, and so is also $\mathbf{r}$. If the selected distribution construct is *box* or *linear*, then the transformed $\mathbf{r}$ is directly the radius around our shape construct. When using other distribution constructs we must allow for other interpretations of $\mathbf{r}$, e.g., the normal distribution, where variance or

σ is more informative. To allow arbitrary distribution constructs, we are using precomputed floating point look up tables, or textures, for this back-mapping into model space.

We are then left with the task of calculating the distribution's σ from the radius $\mathbf{r}$ with one set look up table. The normal distribution has known properties, such as: $X \sim N(\mu, \sigma^2)$ implies $aX+b \sim N(a\mu+b, (a\sigma)^2)$, with a and b as real numbers. As a consequence, we can relate all normal distributed random variables to the standard normal. If $X \sim N(\mu, \sigma^2)$ then

$$Z = \frac{X - \mu}{\sigma} \tag{7}$$

is a standard normal variable $Z \sim N(0, 1)$. Since μ is provided independently by our automatic fitting algorithm we can set $\mu = 0$, and solving Eq. (7) with respect to X, we find that our fitted normal distribution Z scales linearly with the standard normal distribution, and thus we can create the connection between the radius of our normal distribution texture which has a $\sigma = k, k = 5$ (see 3.1) and the σ of our sketched model: $\mathbf{r} = k * \sigma$ we find $\sigma = \mathbf{r}/5$.

We have now described how to extract positions of our abstraction, its distribution width and a scale, based on a given intensity. We will look into more detail on how to apply this information in synthetic and real life cases in the next two sections, but we can already now see the usefulness in cases as pure statistical measures, or quantitative readouts of mean and variance.

4 Visualization and Interaction

From Fig. 3 we find three different visualizations, namely, the data visualization, the prototype illustration, and the residual visualization. We have chosen a 2D KDE as our primary rendering scheme. All rendering of this plot is divided into two stages, the first, evaluating the analytical KDE where a floating point texture is the result, and the second where a color-map, or a height map is used to present this to the user. We call the first step *data rasterization* and the second the visual representation step. Similarly, our models prototypes also share these two steps, one rasterizing step, and one visual representation step, but differently, the model's visual representation is not based on the rasterized result, but on a more condensed format, since the models are inherently easier to abstract, which is why we refer to it as the prototype illustration.

4.1 Visual Representations

To give deeper insights about data, different visualization techniques can be used. By separating the data preparation in rasterization and presentation in visualization, we can achieve a very good flexibility on selection in techniques.

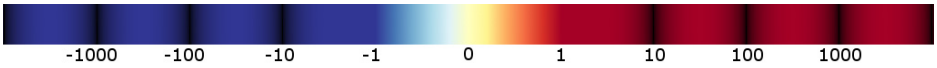


Figure 4: A divergent colorbrewer color-map extended to be continuous and have our "infinite" wave modulation, here logarithmic. Note that within the range defined by the original colormap, no modulation is applied.

The figures throughout this paper shows two major techniques based on the 2D KDE, one using colors, and one using height, both with their strengths and weaknesses. One of the reasons these two methods are chosen, are their ability to show divergent values, i.e., values on both sides of zero. The colored plot achieve this by using a divergent [20] color scheme, and the height map moves its points both up and down.

Over the years, there has been much research on what colors to use for color-maps [13, 20], but also in color theory in general. We do not aim to add anything significant to this field, but we propose a simple technique that shows very good results when coloring floating point textures. In our technique, a small shader is replacing the color-map lookup, introduces iso-contours in either linear steps, or in logarithmic steps, in the range outside of the normal texture. The idea of introducing a waveform into textures to show additional information was first presented by Wong et al. [140]. The technique presented with a logarithmic wave is shown in Fig. 4. In our technique this waveform is never represented in texture memory, but dynamically applied, outside the clamped area in the colormap, by the shader.

4.2 Convergence

When sketching model prototypes, it is inherently hard to accurately or optimally draw the model prototype, as this would require the user to locate a local minimum based on several parameters. As introduced in Sec. 3.2, this would require the user to set five parameters, $\mathbf{p}, \mathbf{r}$, and h, when she/he is modifying a single point model construct. In this work we suggest an assisted fitting prototype, in which the user gets feedback on whether the first suggestion will converge towards his/her desired solution, or not. In the interactive mode, the user moves one point, and when it is released at its new position, the fitting algorithm will initiate. The iterative fitting algorithm is configured such that it will slowly converge/diverge at its first steps. If the initial suggestion was not sufficiently close to the optimal position $\mathbf{x}*$, it will diverge, away from the users desired target position. Instead, we iterate the fitting only a few steps, with constant step size instead of using Newton's method, so that the user can click and redirect the point before it *runs* away. When the user sees that the point is converging towards the desired solution, she/he can initiate the fitting algorithm that will then converge with the speeds that Newton's method offers.

As discussed in Sec. 3.2, we established that we need to compare the least squares of the model vs. the data. To initiate the fitting, we first need a rasterized version of the data. This texture is created once (per frame, or when the dataset is updated), with several data visualization schemes using it concurrently. Next, we need a rasterized version of the model, which we create using a *construct aware* rasterize function, specific for the different implemented models. To recall, we considered the data's rasterized texture as $D(u, v)$ and the models, $M(u, v)$. Next, we need to calculate the least squares, and then we need to sum all the calculations for u and v. These calculations, as specified by Eq. (2), are implemented on a shader, which first performs the least squares, and secondly performs the reduction sum. We discussed the separation of the optimizing previously (see Eq.(6)), and as our tests have shown good convergence, when we iterate stepwise in our previously implied order (ie. one step with position, one with extent, and then with height, before reiterating next step). If we have a point construct, we repeat the process of calculating the sum of least squares, for our position $\mathbf{p}$ in the positions:

$$\mathbf{p} + \Delta u, \quad \mathbf{p} - \Delta u, \quad \mathbf{p} + \Delta v, \quad \mathbf{p} - \Delta v$$

Next, for Newton's method (see Eq. (4)) we need $f'(\mathbf{p}_n)$ and $f''(\mathbf{p}_n)$, now let (for both u and v):

$$f'(\mathbf{p}_n) = \big(s_n(\mathbf{p}_n + \Delta) - s_n(\mathbf{p}_n - \Delta)\big)/2|\Delta|$$
$$f''(\mathbf{p}_n) = \big((s_n(\mathbf{p}_n + \Delta) - s_n(\mathbf{p}_n)) - (s_n(\mathbf{p}_n) - s_n(\mathbf{p}_n - \Delta)) \big)/|\Delta|$$

And thus we are able, to calculate $\mathbf{p}_{n+1} = \mathbf{p}_n - \frac{f'(\mathbf{p}_n)}{f''(\mathbf{p}_n)},\ n \geq 0$.

Next, we fit the extension of our distribution $\mathbf{r}_n$ (in the case of normal distribution, this would be σ), ending up with $\mathbf{r}_{n+1}$, finally, before calculating the scale, or height h_n of the distribution. This we implement in a similar manner, by calculating $s_n(h_n)$, $s_n(h_n + \Delta)$, and $s_n(h_n - \Delta)$, and creating the derivatives, using the same procedure as above, then by Newton's method, we calculate a new height h_{n+1}. To extend the above method, that was defined for point constructs, for piecewise lines, we repeat the first step, finding $\mathbf{p}_{n+1}$ for all $\mathbf{p_n}$ in the piecewise line segment. To impose the constraint on the algorithm, mentioned about degenerating shape constructs, we instead of calculating $f'(\mathbf{p}_n)$ for all directions in u and v, limit movements to those which keeps the distance between points constant. For point $\mathbf{p}_i \in \mathit{linesegment}$, we only allow movements in the direction normal to $\mathbf{n}\perp(\mathbf{p}_{i+1} - \mathbf{p}_{i-1})$

We use a synthetic dataset in order to see how well our technique works on second order features, or features that are hidden under low frequency features with greater intensity. The synthetic function we generated is a 2D version from the one displayed in Fig. 1. This figure shows a primary feature A being the standard normal distribution, and a secondary feature B which is a scaled normal distribution with mean $\mu = 1.0$ and variance $\sigma = 0.2$. In Fig. 2 we show our 2D KDE

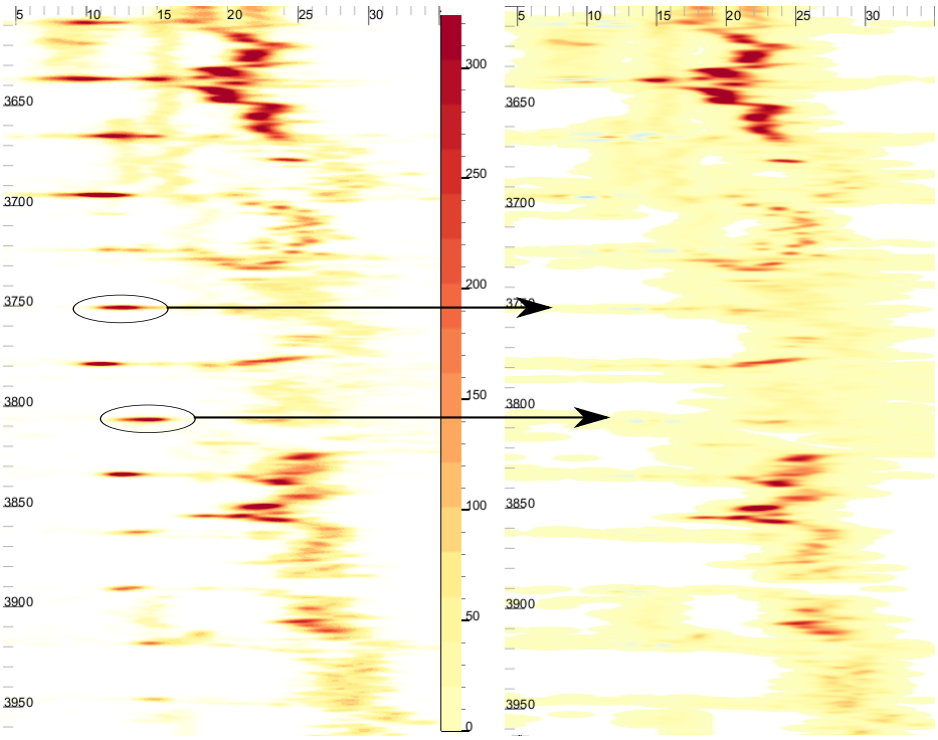


Figure 5: Torque in kN.m over depth. The figure on the left shows the original data, containing some ROB tests we have identified, modeled and subtracted from the residual view to the right.

of the synthetic data, expanded from the 1D version using tensor product distributions. The previous distribution $N(1, 0.2)$ are then $N([1, 1], [0.2, 0.2])$. On the middle and bottom left images it is hard to distinguish the secondary feature, and even with a dynamic color table it is still hard to distinguish. This feature will not separate using iso-curves, since it is placed in a slope (see Fig. 1). The image on the right shows the residual visualization, after our fitting algorithm has placed a model prototype over the primary feature. The primary model identified a feature with mean $\mu = (0.02, 0.01)$ and variance $\sigma = (1.035, 1.04)$, which is close to our original $N([0, 0], [1, 1])$. This subtraction clearly brings the secondary feature to attention. Further prototyping can now be done, and our fitting algorithm now reports the secondary feature to be $N([1.004, 1.004], [0.1989, 0.2015])$ vs. the reference $N([1, 1], [0.2, 0.2])$.

5 Case Study

We now present three case studies, two of process data from the Oil and Gas sector on real-time data generated under drilling operations, and one analyzing the temperature changes through several years.

5.1 Process Data

We will apply our approach to a dataset that contains 116191 time-steps in total, spanning over a period of almost 28 hours, with a varying sample rate from $1/30Hz$ to $30Hz$. It is a multivariate dataset containing 25 variables at each time-step in three major categories, *measured* data from the surface, measurement while drilling (*MWD*) equipment and *derived* data. MWD or down-hole measurements are measured from MWD tools down in the well and then transmitted to the surface via mud pulse.

A prominent usage of these data-streams are logging, early detection and warning in case of incidents or analysis to elaborate on a problem evolving or past. Under drilling operations a fluid, called *mud*, is flowing, from pumps at the rig, through the drill-string exiting at the drill head and then returning to the rig on the outside (*annulus*) of the drill-string. This mud has many functions, e.g., transporting *cuttings*, the rock excavated, to the surface, controling pressure, sealing holes (*permeable formations*), cooling, lubricating, and being the medium for mud pulse telemetry. If a too low mud circulation is used, then we can end up in the situation that more cuttings are generated than what is transported up to the surface. This can lead to dangerous situations where the drill string gets stuck, *stuck pipe*, or *pack off*, where pressure builds very rapidly, leading to *fractured pores*, holes, which in turn can lead to *mud loss*, where mud is lost into the surrounding rock or *formation*. To give an early warning on these incidents, one of the most common strategies applied are friction tests. Increased friction can be a good indicator on gathered cuttings in the hole. To measure friction, two different techniques are applied: torque based tests, and weight based tests. When moving the string up we can expect friction to act as a force against the movement, thus increasing the measured weight, and similarly when moving the string down we expect a lower weight. Rotating the drill string will give a response torque measured on the surface. This torque will increase with increasing friction. This test is called *rotation off bottom* or ROB.

In this study we will look closer into two friction cases, one based on torque, and one based on weight.

Torque Based Friction Estimation

Often it are external contractors, and not the oil companies, that perform the drilling operations, but through the on-shore data transmission, the oil companies

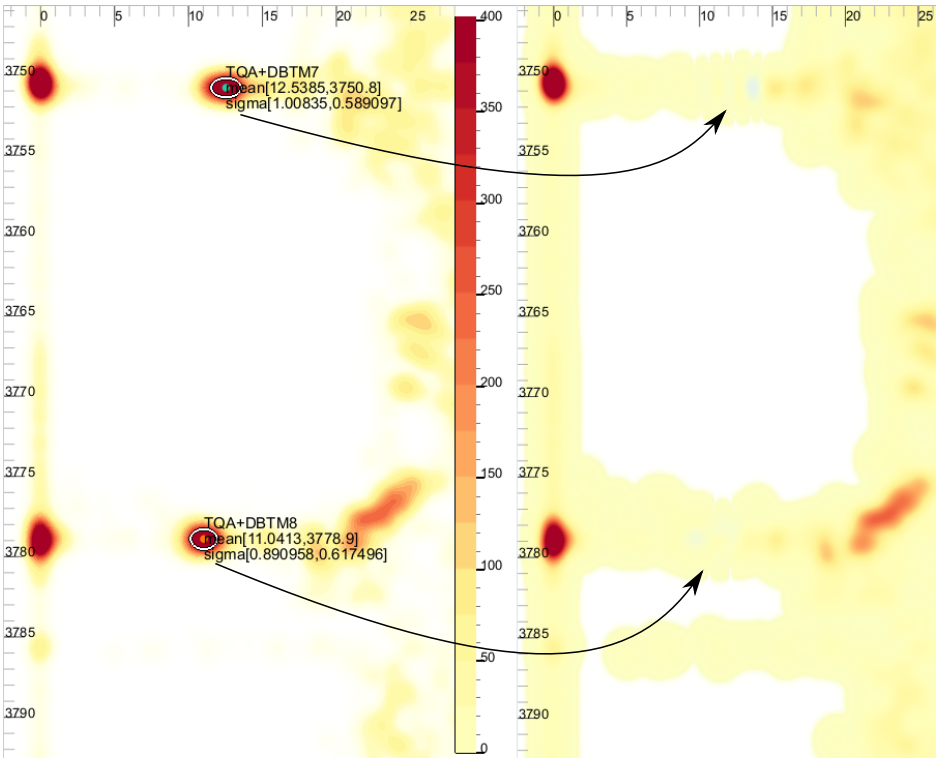


Figure 6: A zoomed in view from Fig. 5 onto two rotation off bottom tests, showing how well this data is modeled and removed from the residual view. The residual view to the right has a larger area of yellow values due to a dynamic range color re-scale.

can monitor what happens off-shore. Also the contractors have to follow certain procedures, and one of these required procedures when drilling is to include regular friction tests. To get comparable results through the well one usually tries to create similar conditions and perform the tests over time, and then select the mean value. This mean value is then compared to simulated expectations, and to previous results. Measuring this value is done on regular intervals. The current work-flow is started by the driller that moves to the depth where the friction test should be performed, corrects pump pressure and then rotates at a constant speed while maintaining the current depth. After noting the torque over the time period a mean is estimated and then sent to personnel responsible to chart the regular results. The rotation tests are performed at depths that correspond to calculations performed before the well is drilled.

Good routines exist for these friction tests, but one problem is that these tests happen only under once an hour, and thus problems can occur between these

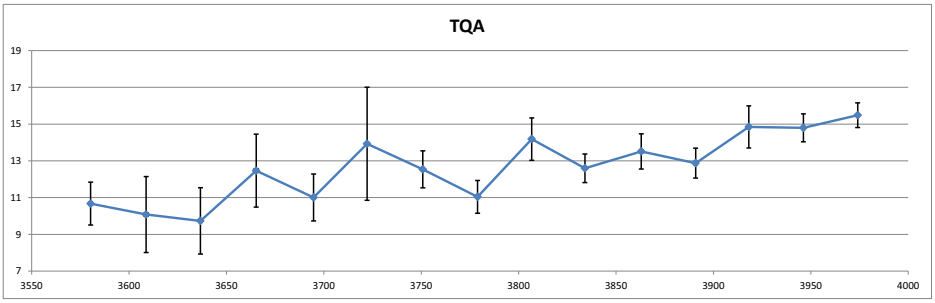


Figure 7: Changing torque in kN.m over depth in feet, for a series of ROB tests. The abstracted results from Fig.5 are shown in a graph with error bars at 1σ for both depth and torque uncertainty. In this figure, the uncertainty for depth is negligable.

tests. We would like to enable the analyst to spot these problems by analysing the streaming data, without interfering with the work done on the rig. The real time data undergoes many different calculations, and presents results based on predefined algorithms, but if the analyst detects something unexpected, and wants to test a hypothesis, she/he is often left without proper tools. Creating new calculations on the data is often out of reach for the analyst, and would take too long if it wasn't. This is where our approach comes into play, by allowing to create and test these hypothesis, and present initial quantified results. In an industry that is more and more dependent on real time data, the idea of rapid prototyping is essential.

Fig.5 shows abstracted and residual data visualization representing ROB. Notice how all the higher densities (representing time spent performing this ROB), is removed in the residual view to the right. Fig.6 shows a zoomed in version of Fig.5, where two of these tests and additionally their quantitative parameters are shown. From this model prototype, one can now read out the average and mean during this ROB. A big advantage our technique has compared to the existing one, where the mean of all measured values during the ROB is taken, is that our technique would show a poor fit, if a poor ROB is performed. An example of a poor ROB is if the samples is increasing, or decreasing during the entire ROB period. Another poor ROB would have its samples clustered at two distinct torques, and a mean would then be misleadingly in middle between them. After matching a total of 15 of these ROB tests in Fig.5 we go to a more abstract view, where all the data is removed, and only the modeled statistcs are left. Fig.7 shows 15 ROB tests, with their standard deviations shown as error bars for both depth and torque. This abstraction illustrates quite well a problematic ROB friction that occurred at 3722 and at 3815 feet, but was put under control at larger depths with more moderate torque and a lower deviation (more certain measurements).

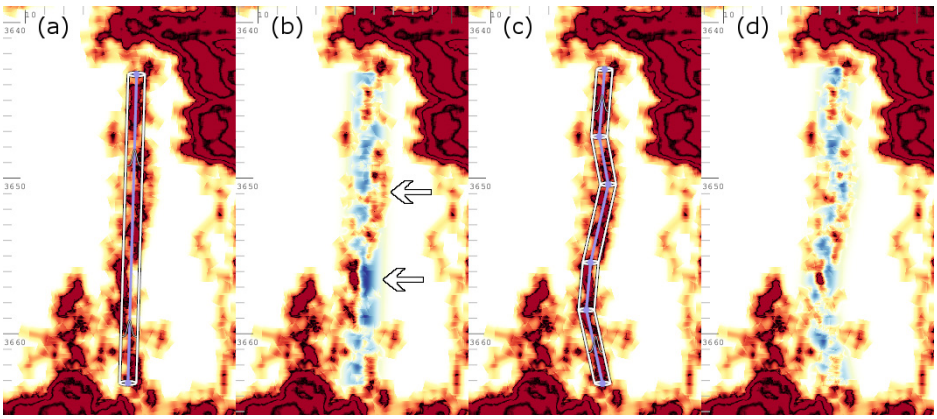


Figure 8: Showing torque over depth. Left to right: (a) A model is sketched and fitted, with the white line showing σ, (b) shows the residual visualization after subtracting (a), arrows pointing to where the model does not match up so well. (c) the model is split up into more line segments and refitted, finally showing a better fit in the residual visualization (d). Using this technique allows the analyst to accurately pinpoint the torque, and thus detect unexpected increasing torque patterns.

Quantifying torque is an important step in knowing how the down-hole conditions are developing, but that is only one step on the way of getting indications on what the friction is. There are no accurate algorithms that exist today, that can accurately calculate friction, even if all different conditions are taken care of. This is the reason why we look at torque, since it is an indication of friction even though many more variables affect the result. If we could pinpoint the similarities of two different reamings, we could at least compare the development of torque, and see if it is within limits of the wellbore trajectory. In fig. 8 we have described such a modeling sequence where we extract torque real time from reaming, and where we then can use the results to compare to the similar reamings, thirty meters up, and down.

Hook Load

Oil rigs use a J-shaped hook to do most of the heavy lifting. This hook weighs several tonnes, and through its cable to the traveling block, it allows measurements of weight. The standard measurement for weight of the string in drilling is exactly this *Hook Load*. Hook load represents the total force pulling down on the hook. This force includes the weight of the drillstring, but is influenced by several factors. Some forces that influence the measured weight include friction along the wellbore wall (especially in deviated wells) and, importantly, buoyant forces on the drillstring caused by its immersion in the drilling fluid. Fig. 9 shows a *Pull Out of Hole* or POOH operation, where the decision has been made to

travel from 3500 meters down in the well, up to the surface, something done every time they need to change a drill bit, or to set a new casing. As expected, the visualization shows that the further up in the well our drill bit is, the drill string gets shorter and thus the total weight is reduced. When measuring the weight when traveling upwards, the measurements are higher than when standing still, and thus we can get a good indication of the friction in the well. This friction is then compared to *expected* values, and we get an indication on the downhole conditions. We bring into the industry, as a novel technique, an approach that allows one to interactively explore this weight, on realtime data streams, calculating mean and variance, continuously through this operation. Fig. 9 shows this usage, on the complete operation. An analyst using this techinque would, looking at this image, first raise some questions on why the operation stopped on several occasions. In this exact operation some unwanted incidents occurred, leading to exactly this non-productive time. The explanation for this finding that this model template describes this exact operation, a POOH, and thus that which does not fit, is not described by the model, and needs further inquiry or modeling. When this was established, this model prototype's values are read out, and compared to the expected values.

5.2 Temperature

In this section we inspect hourly temperature readings from a single weather station, over the course of ten years, courtesy of eKlima [89]. Fig. 10, displays the curve density estimate [27] of these curves over an average year. In this display the most prominent feature is the seasonal change, with high temperatures during the summer, and lower temperatures during the winter. Fig. 10 also features an orange line overlaid to display the cyclic moving average of the yearly temperature for these ten years. Such a moving average is a good representation and abstraction for the yearly temperature, given that the data follows a normal distribution, but might provide false information if not [27]. To investigate how well this average can abstract the data, we first create a new dataset containing the differences between the moving average and the measured temperature. This new dataset of deviations from the moving average, is shown in figure 11. The higher peaks, at approximately zero, during the summer months in this dataset, indicate a more stable temperature, i.e., the average temperature is measured more often. To investigate how well the normal distribution fits the de-trended data, we apply a linear normal-distributed model to it. The resulting difference view between the applied model and the de-trended data is shown in figure 12. Now, as opposed to the previous two figures, the focus is placed on the deviations from the normal distribution. Our attention is drawn to the high intensity above the norm in January and December. Since these intensities are red, they present areas where the measured value is higher represented than the normal distribution. However, since the average is placed at zero, it indicates a "tail" of low

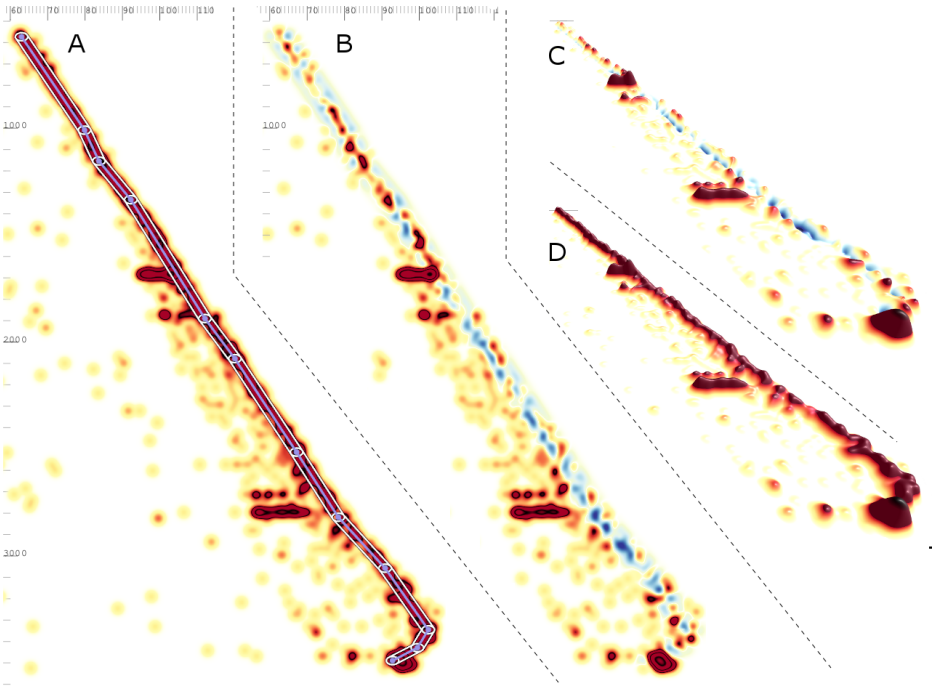


Figure 9: Hook-load in 1000kg (measured weight of the drill string) over depth. The model prototype drawn onto the data (A,D) shows an ideal POOH, and the residual visualization (B,C) shows some anomalies indicating that this operation took longer time than planned. This effective use of our algorithms provides information on the weight (average), trend (slope), confidence on readouts (variance) and also a general how well the data fits the used model template.

temperatures dragging the average down, i.e., a negatively skewed distribution. A second finding here is the anomaly placed mid September, where we find a peak of overrepresented values at an extreme ten degrees below the average. After closer inspection in the dataset, we found that this represents missing values which was defaulted to zero. As a third finding, we point to the light red area above the grey quartile line in the summer months. These indicate that during these months the actual distribution has a positive skewness, leading to a bigger "tail" towards higher temperatures than the average and standard deviation would account for.

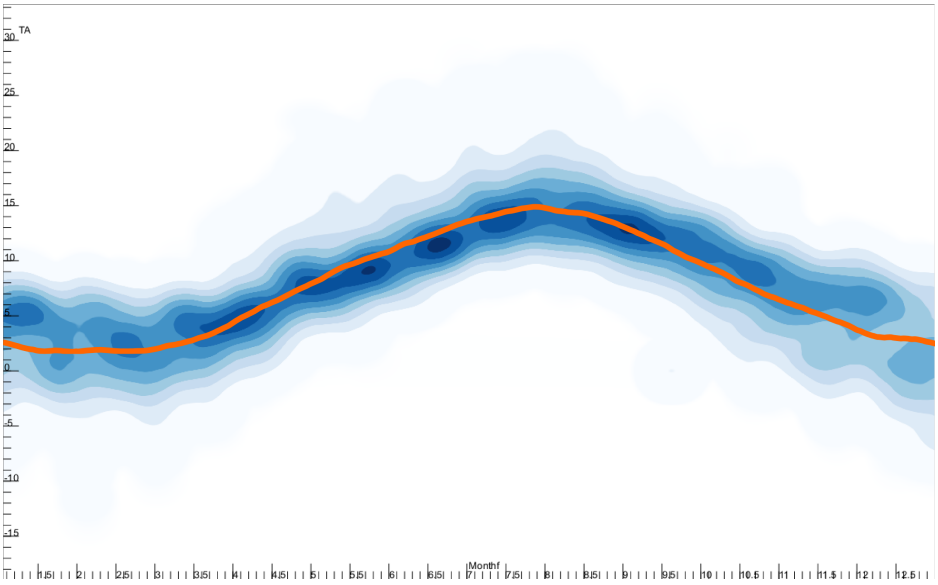


Figure 10: The distribution of measured temperature during an average year based on hourly data for ten years, as calculated by using the curve density estimate [27]. The orange curve shows the cyclic moving average of the temperature over all ten years.

Name	Value	Reference	Pixel err.
Primary μ X	0.02	0	1.7
Primary μ Y	0.01	0	0.85
Primary σ X	1.035	1	2.98
Primary σ Y	1.04	1	3.4
Secondary μ X	1.004	1	0.34
Secondary μ Y	1.004	1	0.34
Secondary σ X	0.1989	0.2	0.0936
Secondary σ Y	0.2015	0.2	0.128

Table 1: Error measurements in data space units and pixels on standard normal distribution with secondary feature, see Fig. 1 and Fig. 2. A rendering with 85 pixels per unit has been used for these calculations.

6 Discussion, Conclusions, and Future Work

Looking at the results from our synthetic test first (section 4.2), we see that not only primary features are properly detected on rasterized data, but also secondary features. In Fig. 2 we can measure that one unit in data space corresponds to 85 pixels, which means that if we can detect, with this exact resolution, close to $1/85 \approx 0.012$ units accuracy, then we have sub pixel accuracy.

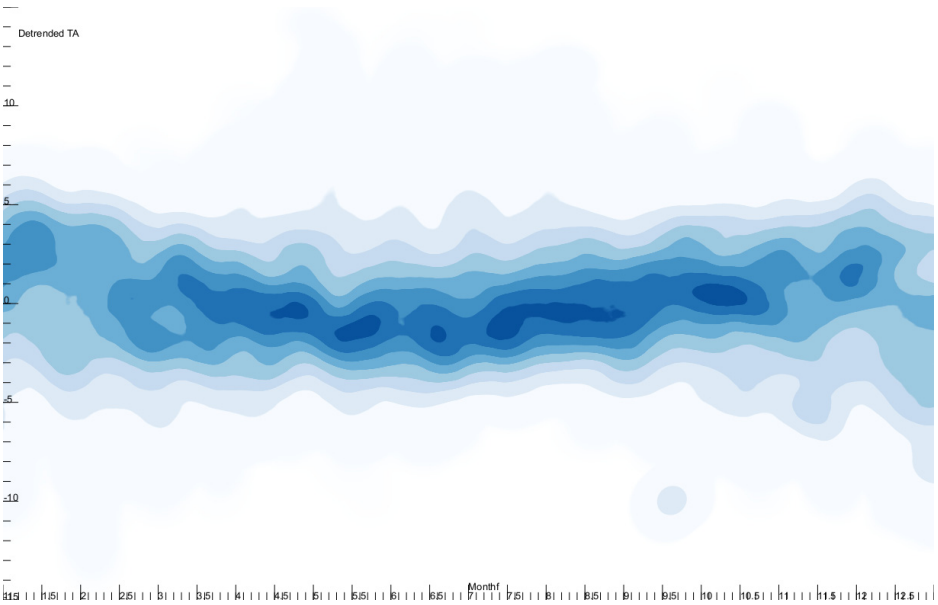


Figure 11: The distribution of temperature deviations from the moving average in Fig. 10, or in other words, seasonally de-trended temperature deviations.

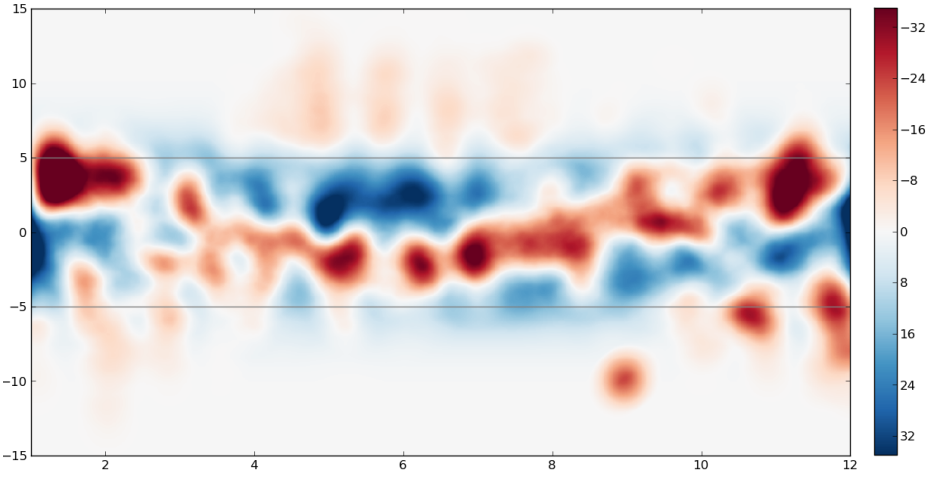


Figure 12: The difference between a linear model and the de-trended data from Fig. 11. The linear model applied has its mean μ on $y = 0$, a $\sigma = 1.85$ and its upper and lower quartile shown as grey lines. Note that due to the diffence view, the deviations shown here are of one magnitude greater than in Fig. 11.

Looking at the results in table 1 we refer to the results on the secondary feature. These results estimate the original model with not only sub-pixel precision, but with at a tenth of a pixel accuracy, on the variance. Further we see that the secondary features mean is estimated to a level of a third of a pixel, also sub pixel precision. On the primary feature, we see that the results are within pixels with regards to the reference, with the estimates a little on the high side. The primary feature is the first one fitted in our data visualization, and thus it includes the secondary feature in its estimate, something that can explain the pixel offset in μ. The results on accuracy are very promising, which is very interesting considering that our approach achieves these results at $O(n)$ (we rasterize n points once).

An important characteristic of our approach is the high degree of interactivity. When displaying streaming data, it is important to have a visualization scheme that is able to handle large time windows, i.e., if data is streamed one needs at some point to either omit "old" data from the visualization, or support a multi resolution scheme. We have implemented visualization mappings that allow fast rendering (> 60 fps), even if we show datasets spanning several days ($> 200k$ samples). The feedback on convergence, (or divergence), is also an important aspect that facilitates interactive analysis.

With this work we have demonstrated how data visualization can benefit from interactive model prototyping, externalization and subtraction so that expert users can rapidly proceed through an in depth analysis of streaming process data, following the *visualize and observe, sketch and fit, externalize and subtract, then iterate* pattern. Subtracting identified features from the data visualization allows the user to reveal secondary features and additionally results in an externalized prototype giving quantification and overview. While the usefulness of this approach might be obvious in the synthetic cases, we have also confirmed this on process data from the oil and gas sector.

We have shown that interactive model prototyping in visualization space can accurately quantify measured data. Moreover, we have shown that an analyst can quickly compare suggestions for formal models, by bringing them into the visualization, perform prototyping, and get quantitative results on how well they fit. Another important part of our work has been to move visualizations beyond the initial discovery, and to give the users a view into secondary features. A general conclusion from our work is that application processes usually don't stop after discoveries in visualization and that is therefore important for visualization research to more intensely think about what has to follow visualization, e.g., externalization, quantification and ultimately action.

In future work, we plan to look further into different reconstruction techniques, and also different distributions. An interesting aspect would be to investigate the support for distributions with rotations or shear, by enabling support for a full covariance matrix, instead of the vector $\mathbf{r}$. Likewise, extending the support of normal distributions to also quantify skewness, or even kurtosis, would perhaps be an interesting path. Since we are utilizing floating point textures as interface

to the distributions, we estimate that it will be relatively easy to implement different distribution constructs and templates. Another plan is to extend the piecewise linear model to support higher-order templates, like spline curves. It would interesting to consider an extension to multivariate fields, or even to three dimensional fields, using 3D rasterizing functions.

7 Acknowledgements

The work presented here is a part of the project "e-Centre Laboratory for Automated Drilling Processes" (eLAD), participated by International Research Institute of Stavanger, Christian Michelsen Research and Institute for Energy Technology. The eLAD project is funded by grants from the Research Council of Norway (Petromaks Project 176018/S30, 2007-2011), Statoil ASA and ConocoPhillips Norway.

Bibliography

[1] C. Ahlberg. Spotfire: an information exploration environment. *SIGMOD Record*, 25:25–29, 1996.

[2] W. Aigner, S. Miksch, W. Müller, H. Schumann, and C. Tominski. Visualizing time-oriented data: A systematic view. *Computers & Graphics*, 31(3):401–409, 2007.

[3] G. Andrienko, N. Andrienko, J. Dykes, S. Fabrikant, and M. Wachowicz. Geovisualization of dynamics, movement and change: key issues and developing approaches in visualization research. *Information Visualization*, 7(3):173–180, 2008.

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

[5] N. Andrienko and G. Andrienko. Spatial generalization and aggregation of massive movement data. *IEEE Trans. Visualization and Computer Graphics*, 2010. (RapidPost).

[6] A. Artero, M. de Oliveira, and H. Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. *Proc. of IEEE Symp. on INFOVIS*, 2004.

[7] Asa data expo 2009. http://stat-computing.org/dataexpo/2009. [Online; accessed Nov-2010].

[8] Audacity. The Free Audio Editor and Recorder. audacity.sourceforge.net. [Online; accessed Nov-2010].

[9] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Trans. Visualization and Computer Graphics (Vis 2008)*, 14(6):1428–1435, 2008.

[10] R. Bade, S. Schlechtweg, and S. Miksch. Connecting time-oriented data and information to a coherent interactive visualization. *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 105–112, 2004.

[11] A. H. Barr. Global and local deformations of solid primitives. *Siggraph Comp. Graph.*, 18(3):21–30, 1984.

[12] A. V. Bartrolí, R. Wegenkittl, A. König, and E. Gröller. Nonlinear virtual colon unfolding. In *Proc. IEEE Vis.*, pages 411–420, 2001.

[13] L. Bergman, B. Rogowitz, and L. Treinish. A rule-based tool for assisting colormap selection. In *Proc. Visualization*, pages 118–125, 1995.

[14] L. Berry and T. Munzner. BinX: Dynamic exploration of time series datasets across aggregation levels. In *Proc. IEEE Symp. Information Visualization (InfoVis 2004)*, pages 215–216, 2004.

[15] Z. I. Botev. A novel nonparametric density estimator. *Postgrad. Sem. Series, Math. , The Univ. of Queensland*, 2006.

[16] B. W. Boyle, R. Madhavan, and J. Jundt. Wired pipe joint with current-loop inductive couplers, 2003. Patent No.: US 6641434.

[17] A. Braseth, V. Nurmilaukas, and J. Laarni. Realizing the information rich design for the loviisa nuclear power plant. *American Nuclear Society International Topical Meeting on Nuclear Plant Instrumentation, Control, and Human-Machine Interface Technologies (NPIC&HMIT), 6*, 2009.

[18] A. Braseth, Ø. Veland, and R. 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, Columbus, Ohio*, 2004.

[19] C. Brewer. Color use guidelines for data representation. In *Proc. Section on Statistical Graphics*, pages 55–60, 1999.

[20] C. A. Brewer. Color advice for maps. http://ColorBrewer.org.

[21] M. D. Buhmann. *Radial basis functions*. Cambridge Uni. Press, 2003.

[22] M. Chen, C. Correa, S. Islam, M. W. Jones, P.-Y. Shen, D. Silver, S. J. Walton, and P. J. Willis. Manipulating, Deforming and Animating Sampled Object Representations. *Comp. Graph. For.*, 26(4):824–852, 2007.

[23] M. Chen, D. Silver, A. S. Winter, V. Singh, and N. Cornea. Spatial transfer functions: a unified approach to specifying deformation in volume modeling and animation. In *Proc. Vol. Graph.*, pages 35–44. ACM, 2003.

[24] R. Chin and C. Dyer. Model-based recognition in robot vision. *ACM Computing Surveys (CSUR)*, 18(1):67–108, 1986.

[25] C. Correa, D. Silver, and M. Chen. Feature aligned volume manipulation for illustration and visualization. *IEEE TVCG*, 12(5):1069–1076, 2006.

[26] R. Crawfis and N. Max. Texture splats for 3d scalar and vector field visualization. In *Proc. IEEE Visualization Conf. (Vis '93)*, pages 261–266, Oct 1993.

[27] O. Daae Lampe and H. Hauser. Curve density estimates. *Computer Graphics Forum*, 30(3):633–642, 2011.

[28] O. Daae Lampe and H. Hauser. Interactive visualization of streaming data with kernel density estimation. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis 2011)*, pages 171–178, March 2011.

[29] O. Daae Lampe, J. Kehrer, and H. Hauser. Visual analysis of multivariate movement data using interactive difference views. In *Proceedings of Vision, Modeling, and Visualization (VMV 2010)*, pages 315–322, 2010.

[30] M. desJardins and P. Rheingans. Visualization of high-dimensional model characteristics. In *Workshop on New Paradigms in Information Visualization and Manipulation*, pages 6–8, 1999.

[31] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. Eurographics/IEEE-TCVG Symp. on Visualization (VisSym 2003)*, pages 239–248, 2003.

[32] J. Duchon. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. *Constructive theory of functions of several variables*, pages 85–100, 1977.

[33] S. G. Eick and G. J. Wills. High interaction graphics. *European Journal of Operations Research*, 81(3):445–459, 1995.

[34] G. Ellis and A. J. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE TVCG*, 13(6), 2007.

[35] M. Ericson. Keynote: Visualizing Data for the Masses: Information Graphics at The New York Times. *VisWeek*, 2007.

[36] Federal election commission. http://www.fec.gov/.

[37] J. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proc. of IEEE Symp. on INFOVIS*, 2002.

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

[39] D. Fisher. Hotmap: Looking at geographic attention. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1184–1191, 2007.

[40] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Ann. Eugenics 7*, 1936. StatLib http://lib.stat.cmu.edu/.

[41] M. Florek and H. Hauser. Quantitative data visualization with interactive kde surfaces. In *Proceedings of the Spring Conference on Computer Graphics (SCCG 2010)*, pages –, May 2010.

[42] M. Florek and H. Hauser. Interactive bivariate mode trees for visual structure analysis. In *Proceedings of the Spring Conference on Computer Graphics (SCCG 2011)*, pages –, 2011.

[43] F. Frenet. Sur les courbes à double courbure. *Journal des Mathematiques Pures et Appliquees*, 17:437–447, 1852.

[44] R. Frigg and S. Hartmann. "models in science", the stanford encyclopedia of philosophy (summer 2009 edition), edward n. zalta (ed.). http://plato.stanford.edu/archives/sum2009/entries/models-science.

[45] R. Fuchs and H. Hauser. Visualization of multi-variate scientific data. *Computer Graphics Forum*, 28(6):1670–1690, 2009.

[46] J. Gain and D. Bechmann. A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph.*, 27(4):1–21, 2008.

[47] V. D. Gesù and V. V. Starovoitov. Distance-based functions for image comparison. *Pattern Recognition Letters*, 20(2):207–214, 1999.

[48] A. Gray and A. Moore. Nonparametric density estimation: Toward computational tractability. In *SIAM Int. Conf. on Data Mining*, 2003.

[49] E. Gröller. Nonlinear ray tracing: Visualizing strange worlds. *The Visual Computer*, 11(5):263–274, 1995.

[50] E. Grundy, M. Jones, R. Laramee, R. Wilson, and E. Shepard. Visualisation of sensor data from animal movement. *Computer Graphics Forum*, 28(3):815–822, 2009.

[51] S. Haker, S. Angenent, A. Tannenbaum, and R. Kikinis. Non-distorting flattening for virtual colonoscopy. In *MICCAI*, pages 358–366, 2000.

[52] M. Hao, D. Keim, U. Dayal, D. Oelke, and C. Tremblay. Density Displays for Data Stream Monitoring. *Computer Graphics Forum*, 27(3):895–902, 2008.

[53] M. Hao, D. Keim, U. Dayal, and T. Schreck. Multi-resolution techniques for visual exploration of large time-series data. In *EuroVis 2007*, pages 27–34, 2007.

[54] M. A. Harrower and C. A. Brewer. ColorBrewer.org: An Online Tool for Selecting Color Schemes for Maps. *The Cartograpic Journal*, 40(1):27–37, 2003.

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

[56] S. He, R. Dai, B. Lu, C. Cao, H. Bai, and B. Jing. Medial axis reformation: A new visualization method for ct angiography. *Academic Radiology*, 8:726–733, 2001.

[57] H. Hochheiser and B. Shneiderman. Dynamic query tools for time series data sets: timebox widgets for interactive exploration. *Proc. IEEE Symp. Information Visualization (InfoVis 2004)*, 3(1):1–18, 2004.

[58] W. Hong, X. Gu, F. Qiu, M. Jin, and A. Kaufman. Conformal virtual colon flattening. In *Symp. Solid Phys. Mod.*, pages 85–93. ACM, 2006.

[59] J. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[60] C. Hurter, B. Tissoires, and S. Conversy. FromDaDy: spreading aircraft trajectories across views to support iterative queries. *IEEE Trans. Visualization and Computer Graphics*, 15:1017–1024, 2009.

[61] Y. Jang, M. Weiler, M. Hopf, J. Huang, D. Ebert, K. Gaither, and T. Ertl. Interactively visualizing procedurally encoded scalar fields. In *Proc. of EG/IEEE TCVG Symp. on Vis. VisSym*, volume 4, 2004.

[62] T. Jankun-Kelly and K.-L. Ma. Visualization exploration and encapsulation via a spreadsheet-like interface. *IEEE Trans. Visualization and Computer Graphics*, 7(3):275–287, 2001.

[63] F. Janoos, S. Singh, O. Irfanoglu, R. Machiraju, and R. Parent. Activity analysis using spatio-temporal trajectory volumes in surveillance applications. In *Proc. IEEE VAST*, pages 3–10, 2007.

[64] D. Jerding and J. Stasko. The information mural: A technique for displaying and navigating large information spaces. *Proc. IEEE Visualization Conf. (Vis 2002)*, 4(3):257–271, 2002.

[65] M. Jern and J. Franzén. GeoAnalytics—exploring spatio-temporal and multivariate data. In *Proc. Int'l. Conf. Information Visualization (IV '06)*, pages 25–31, 2006.

[66] J. Johansson, P. Ljung, and M. Cooper. Depth cues and density in temporal parallel coordinates. *Proceedings of Eurographics/IEEE-VGTC Symposium on Visualization*, 7:35–42, 2007.

[67] E. Jones, T. Oliphant, P. Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[68] A. Kanitsar, D. Fleischmann, R. Wegenkittl, P. Felkel, and M. E. Gröller. CPR - Curved Planar Reformation. *Proc. IEEE Vis.*, 0:37–44, 2002.

[69] A. Kanitsar, R. Wegenkittl, D. Fleischmann, and M. Gröller. Advanced curved planar reformation: flattening of vascular structures. *Proc. IEEE Vis.*, pages 43–50, Oct. 2003.

[70] J. Kehrer, P. Filzmoser, and H. Hauser. Brushing moments in interactive visual analysis. *Computer Graphics Forum*, 29(3):813–822, 2010.

[71] D. Keim, F. Mansmann, J. Schneidewind, J. Thomas, and H. Ziegler. Visual analytics: Scope and challenges. *Visual Data Mining*, pages 76–90, 2008.

[72] D. Keim, F. Mansmann, J. Schneidewind, and H. Ziegler. Challenges in visual data analysis. In *Proc. Int'l. Conf. Information Visualization (IV '06)*, pages 9–16, 2006.

[73] P. Kidwell, G. Lebanon, and W. Cleveland. Visualizing Incomplete and Partially Ranked Data. *IEEE TVCG*, 14(6), 2008.

[74] R. Kincaid. SignalLens: Focus+context applied to electronic time series. *IEEE Trans. Visualization and Computer Graphics (Vis 2010)*, 16(6):900–907, 2010.

[75] F. Klok. Two moving coordinate frames for sweeping along a 3d trajectory. *Computer Aided Geometric Design*, 3(3):217 – 229, 1986.

[76] J. Kniss, S. Premoze, M. Ikits, A. Lefohn, C. Hansen, and E. Praun. Gaussian transfer functions for multi-field volume visualization. In *Proc. IEEE Visualization Conf. (Vis 2003)*, pages 497–504, 2003.

[77] R. Kosara, F. Bendix, and H. Hauser. Timehistograms for large, time-dependent data. *Joint Eurographics–IEEE TCVG Symposium on Visualization*, 2004.

[78] Y. Kurzion and R. Yagel. Space deformation using ray deflectors. In *6th Eurographics Workshop on Rendering 95*, pages 21–32, 1995.

[79] N. Lee and M. Rasch. Tangential curved planar reformation for topological and orientation invariant visualization of vascular trees. *IEEE Eng. in Med. and Bio. Soc.*, pages 1073–1076, 2006.

[80] Z. Liu and J. Stasko. Mental Models, Visual Reasoning and Interaction in Information Visualization: A Top-down Perspective. *IEEE Trans. Visualization and Computer Graphics*, 16(6):999–1008, 2010.

[81] H. Löffelmann and E. Gröller. Ray Tracing with Extended Cameras. *Journal of Visualization and Computer Animation*, 7(4):211–228, 1996.

[82] H. Löffelmann, T. Kučera, and E. Gröller. Visualizing Poincaré maps together with the underlying flow. In *Mathematical Visualization - Algorithms, Applications and numerics*, pages 315–328. Springer, 1998.

[83] K. Matkovic, W. Freiler, D. Gracanin, and H. Hauser. Comvis: a coordinated multiple views system for prototyping new visualization technology. In *Proceedings of the 12th International Conference Information Visualisation*, 7 2008.

[84] S. Miksch, A. Seyfang, W. Horn, and C. Popow. Abstracting steady qualitative descriptions over time from noisy, high-frequency data. *AIMDM '99 Proceedings of the Joint European Conference on Artificial Intelligence in Medicine and Medical Decision Making*, pages 281–290, 1999.

[85] M. Minnotte, D. Marchette, and E. Wegman. New terrain in the mode forest. *COMPUTING SCIENCE AND STATISTICS*, pages 473–477, 1998.

[86] M. C. Minnotte and D. W. Scott. The mode tree: a tool for visualization of nonparametric density features. *Journal of Computational and Graphical Statistics*, 2, 1993.

[87] P. Muigg, J. Kehrer, S. Oeltze, H. Piringer, H. Doleisch, B. Preim, and H. Hauser. A Four-level Focus+Context Approach to Interactive Visual Analysis of Temporal Features in Large Scientific Data. *Computer Graphics Forum*, 27(3):775–782, 2008.

[88] J. Nocedal and S. Wright. *Numerical Optimization*. Springer, 1999.

[89] Norwegian Meteorological Institute. eKlima. eklima.met.no. [Online; accessed Nov-2010].

[90] M. Novotný and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE TVCG*, 12(5), 2006.

[91] G. Nygaard. elad. http://goo.gl/OTkXK. Accessed 01.09.2011.

[92] H. Pagendarm and F. Post. *Comparative Visualization: Approaches and Examples*. Delft University of Technology, Faculty of Technical Mathematics and Informatics, 1995.

[93] E. Parzen. On estimation of a probability density function and mode. *The Annals of Mathematical Statistics*, 33(3), 1962.

[94] F. H. Post, B. Vrolijk, H. Hauser, R. S. Laramee, and H. Doleisch. The State of the Art in Flow Visualization: Feature Extraction and Tracking. *Computer Graphics Forum*, 22:775–792, 2003.

[95] E. Ramos and D. Donoho. 1983 ASA data exposition dataset. http://lib.stat.cmu.edu/datasets/.

[96] J. Rasmussen. Skills, rules, knowledge; signals, signs, and symbols, and other distinctions in human performance models. *IEEE Transactions on Systems, Man and Cybernetics*, 13:257–266, 1983.

[97] J. Rasmussen. The role of hierarchical knowledge representation in decisionmaking and system management. *IEEE Transactions on Systems, Man, & Cybernetics*, 15(2):234–243, 1985.

[98] H. Reijner. The development of the horizon graph. In *IEEE Visualization Workshop: From Theory to Practice: Design, Vision and Visualization*, 2008.

[99] C. Rezk-Salama, M. Scheuering, G. Soza, and G. Greiner. Fast volumetric deformation on general purpose hardware. In *Proc. on Workshop on Graph. Hardware*, pages 17–24. ACM, 2001.

[100] P. Rheingans and M. desJardins. Visualizing high-dimensional predicitive model quality. In *IEEE Visualization*, pages 493–496, 2000.

[101] M. Rosenblatt. Remarks on some nonparametric estimates of a density function. *The Annals of Mathematical Statistics*, 27(3), 1956.

[102] T. Saito, H. N. Miyamura, M. Yamamoto, H. Saito, Y. Hoshiya, and T. Kaseda. Two-tone pseudo coloring: Compact visualization for one-dimensional data. *Proc. IEEE Symp. Information Visualization (InfoVis 2005)*, pages 173–180, 2005.

[103] R. Scheepens, N. Willems, H. van de Wetering, , and J. J. van Wijk. Interactive visualization of multivariate trajectory data with density maps. In *Proceedings of the IEEE Pacific Visualization Symposium (PacificVis 2011)*, pages 147–15, March 2011.

[104] D. W. Scott. *Multivariate density estimation: theory, practice, and visualization*. Wiley-Interscience, illustrated edition, 1992.

[105] T. W. Sederberg and S. R. Parry. Free-form deformation of solid geometric models. In *Proc. Siggraph '86*, pages 151–160. ACM, 1986.

[106] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *Proc. IEEE Symp. Visual Languages*, pages 336–343, 1996.

[107] Y. B. Shrinivasan and J. J. van Wijk. Supporting the analytical reasoning process in information visualization. In *CHI '08: Proc. of SIGCHI on Human factors in computing systems*, pages 1237–1246, 2008.

[108] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

[109] W. Silvert. Modelling as a discipline. *International Journal of General Systems*, 30(3):261–282, 2001.

[110] K. Singh and E. Fiume. Wires: a geometric deformation technique. In *Proc. Comp. Graph. and Interactive Tech.*, pages 405–414. ACM, 1998.

[111] J. P. Snyder. *Flattening the Earth: Two Thousand Years of Map Projections*. University of Chicago Press, 1993.

[112] C. Stolte, D. Tang, and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *IEEE Trans. Visualization and Computer Graphics*, 8(1):52–65, 2002.

[113] R. W. Sumner, J. Schmid, and M. Pauly. Embedded deformation for shape manipulation. *ACM Trans. Graph.*, 26(3):80, 2007.

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

[115] Tableau. http://www.tableausoftware.com/. Accessed 01.09.2011.

[116] D. Tarn. An introduction to kernel density estimation. http://school.maths.uwa.edu.au/~duongt/seminars/intro2kde/, 2001.

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

[118] T. J. True and J. F. Hughes. Volume warping. In *Proc. IEEE Vis.*, pages 308–315, 1992.

[119] E. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, 1997.

[120] E. Tufte. *Beautiful evidence*, volume 23. Graphics Press Cheshire, CT, 2006.

[121] E. R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1983.

[122] E. R. Tufte. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Graphics Press, 1997.

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

[124] B. A. Turlach. Bandwidth Selection in Kernel Density Estimation: A Review. In *CORE and Institut de Statistique*, 1993.

[125] L. A. Tweedie, R. Spence, D. Williams, and R. Bhogal. The attribute explorer. In *In Proc. of the Video Track of the ACM Conference on Human Factors in Computing Systems*, pages 435–436, 1994.

[126] V. Verma and A. Pang. Comparative flow visualization. *IEEE Trans. Visualization and Computer Graphics*, 10(6):609–624, 2004.

[127] K. Vicente and J. Rasmussen. Ecological interface design: theoretical foundations. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(4):589 –606, jul/aug 1992.

[128] T. Vrtovec, B. Likar, and F. Pernus. Automated curved planar reformation of 3D spine images. *Physics in Medicine and Biology*, 50(19):4527, 2005.

[129] D. F. Walnut. *An Introduction to Wavelet Analysis*. Springer, 2004.

[130] M. Wand and M. Jones. *Kernel Smoothing*. Monographs on Statistics and Applied Probability 60. Chapman & Hall, 1995.

[131] G. Wang, G. McFarland, B. Brown, and M. Vannier. GI tract unraveling with curved cross sections. *IEEE Trans. Med. Img.*, 17(2):318–322, 1998.

[132] M. Ward. XmdvTool: Integrating multiple methods for visualizing multivariate data. In *Proc. IEEE Visualization Conf. (Vis '94)*, pages 326–336, 1994.

[133] R. Westermann and C. Rezk-Salama. Real-time volume deformations. *Comp. Graph. Forum*, 20(3):443–451, 2001.

[134] U. Weyer, A. Braseth, M. Eikås, L. Hurlen, and P. Kristiansen. Safety presentation in large screen displays - a new approach. *SPE Intelligent Energy Conference and Exhibition*, 2010.

[135] G. Whittaker and D. Scott. Nonparametric regression for analysis of complex surveys and geographic visualization. *Sankhyā: The Indian Journal of Statistics, Series B*, 1999.

[136] N. Willems, H. van de Wetering, and J. van Wijk. Visualization of vessel movements. *Computer Graphics Forum*, 28(3):959–966, 2009.

[137] D. Williams, S. Grimm, E. Coto, A. Roudsari, and H. Hatzakis. Volumetric curved planar reformation for virtual endoscopy. *IEEE TVCG*, 14(1):109–119, 2008.

[138] M. Wohlfart and H. Hauser. Story Telling for Presentation in Volume Visualization. In *EuroVis*, 2007.

[139] P. C. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. Dynamic visualization of transient data streams. *Information Visualization, IEEE Symposium on*, pages 97–104, 2003.

[140] P. C. Wong, H. Foote, D. L. Kao, L. R. Leung, and J. Thomas. Multivariate visualization with data fusion. *Information Visualization*, 1, 2002.

[141] J. Woodring and H.-W. Shen. Multi-variate, time-varying, and comparative visualization with contextual cues. *IEEE Trans. Visualization and Computer Graphics*, 12:909–916, 2006.

[142] D. Yang, Z. Xie, E. A. Rundensteiner, and M. O. Ward. Managing discoveries in the visual analytics proc. *SIGKDD Explor. Newsl.*, 9(2), 2007.

[143] J. S. Yi, Y. A. Kang, J. Stasko, and J. Jacko. Toward a deeper understanding of the role of interaction in information visualization. *IEEE Trans. Visualization and Computer Graphics*, 13(6):1224–1231, 2007.

[144] S. Zambal, A. Schöllhuber, K. Bühler, and J. Hladuvka. Fast and robust localization of the heart in cardiac MRI series. *Proc. of Int. Conf. on Computer Vision Theory and Applications*, 2008.

[145] A. Zhou, Z. Cai, L. Wei, and W. Qian. M-kernel merging: Towards density estimation over data streams. *IEEE Proceedings of the Eighth International Conference on Database Systems for Advanced Applications (DASFAA'03)*, 2003.