8-2009

Enhancing User Queries in Scientific Visualization with Distribution Information

Christopher Johnson
University of Tennessee - Knoxville

To the Graduate Council:

I am submitting herewith a dissertation written by Christopher Johnson entitled "Enhancing User Queries in Scientific Visualization with Distribution Information." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Jian Huang, Major Professor

We have read this dissertation and recommend its acceptance:

Michael Thomason, Bradley Vander Zanden, Shih-Lung Shaw

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

To the Graduate Council:

I am submitting herewith a dissertation written by Christopher Johnson entitled "Enhancing User Queries in Scientific Visualization with Distribution Information." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Computer Science.

Jian Huang, Major Professor

We have read this dissertation
and recommend its acceptance:

Michael Thomason

Bradley Vander Zanden

Shih-Lung Shaw

Accepted for the Council:

Carolyn R. Hodges

Vice Provost and Dean of the
Graduate School

(Original signatures are on file with official student records.)

Enhancing Query-driven Visualization with Distribution Information

A Dissertation
Presented for the
Doctor of Philosophy
Degree
The University of Tennessee, Knoxville

Christopher Johnson
August 2009

Acknowledgements

I thank my advisor Dr. Jian Huang for getting me started on the path of visualization during a computer graphics course. Armed with lots of patience and insight, Dr. Huang guided me throughout my years at UTK to think critically, labor with revision in mind, and communicate directly. I am also thankful for the communities at Oak Ridge National Laboratory, the Ames Laboratory, and the Department of Computer Science at Iowa State University who supported me in pursuit of my degree and gave me much freedom to learn and code.

I am especially grateful for my wife Cedar, whose unflagging companionship sustained me through uncertain times. Her selfless serving, love of stories, and dreams of a simple future made completing my graduate studies a more worthwhile task. I also thank my infant son Lewis for confirming my career choice with his love of screensavers.

Abstract

Scientific visualization is concerned with the graphical portrayal of data. Using symbols, color, and natural perceptual cues, humans gain insight into collections of raw numbers that may not be as efficiently processed in non-graphical formats. For simple data, visualization may require only a simple mapping between numeric values and a color scale. But modern scientific, economic, and social data is far from simple. Multiple variables, duration of time, fine resolutions, and wide sampling have yielded data sets of unprecedented complexity. The mapping between such data and its visual appearance is difficult to define. The works described here attempt to make defining this mapping more manageable to users of visualization software.

We describe three tools to assist users in finding features of interest in familiar, probabilistic terms. The first tool is an expressive query language in which the user can describe features using criteria based on the frequency distribution of values in local neighborhoods. The query language is closely integrated with the visualization to provide meaningful insight into the matching data and feedback to guide modification of query parameters. The second tool targets the investigation of high-dimensional aspatial data. A common technique called *parallel coordinates* is extended to three dimensions in order to make relationships between variables more apparent. A semiautomatic method of finding compelling viewpoints of the data in 3-D space is introduced. The user defines what features are compelling in terms of a view's image space distribution of color and depth values. Lastly, we describe a novel frequency histogram called a *periograph* for displaying cyclic temporal data. Through periographs, users inspect seasonal trends and their variations through time.

Contents

List of Tables

List of Figures

Introduction

> To judge what one must do to obtain a good or avoid an evil, it is necessary to consider not only the good and the evil themselves, but also the probability that they happen, or not; and to view geometrically the proportion that all these things have together.
>
> Antoine Arnauld and Pierre Nicole, *Port-Royal Logic*, 1662.

Motivation

A key step in scientific study is the location and identification of features within data. For all the measurements recorded, meaningful labels must be assigned to the data so that it can be understood. Starting with high-level notions about phenomena, scientists wonder where and under what conditions these phenomena occur. It is at this high level that life-impacting decisions are made, so computational tools that perform fast and flexible feature querying are crucial to the scientific process.

Visualization is an ideal medium for presenting features for high-level interpretation. By expressing dense and complex data through a visual representation, we leverage the natural abilities of our human perceptual system to assess our environment. To effectively connect low-level data to a human viewer, visualization must find solutions to several challenges—namely, it must: 1) clearly represent numeric measurements or qualitative descriptors in a visually interpretable format, 2) deemphasize irrelevant data while drawing focus to important data, and 3) offer the user some interactive means of arbitrating between important and unimportant data.

A common example of applying visualization to study features is radiologists' use of three-dimensional computed tomography (CT) data sets. Within these volumes of data, the density of atoms in human patients has been recorded at discrete locations. Differing tissue types like skin, muscle, and fat have distinct density profiles, and by assigning different colors and opacity according to locations' densities, the tissues can be visually distinguished when rendered. Radiologists analyze the structure and arrangement of the different tissues to diagnose patients' conditions. If a particular tissue feature is of interest, the associated range of densities is given non-zero opacity and the remaining densities are made transparent.

Despite its importance, classifying data into features is extremely difficult for real-world data sets. Materials intermix and overlap, measured values map ambiguously back to their physical sources (e.g., tissue densities are not unique), and many features of the data cannot be described by a single scalar value. Additionally, data sets are continually growing in extent and resolution as acquisition methods improve, making computationally efficient processing of data a necessity. Visualization is entering new data-rich domains where conventional classification methods fail. Since so much of the scientific process rests on the foundation of being able to identify phenomena, it is essential that improved classification and query methods be developed.

To facilitate more powerful feature visualization for data sets of any dimensionality or structure, we propose in this work several methods for incorporating data sets' frequency distributions into the user query and classification process. Rather than rely on a single density value to determine a tissue's color, for example, through these contributions radiologists can leverage the frequency distribution to understand the tissue in its context. Instead of requiring the user to navigating through time and space trying to find features manually, features can be described and detected in terms of distribution metrics. The strengths of distributions are their generality and simplicity compared to a raw data set, and reasoning based on frequencies is very natural to users, with histograms appearing throughout media, corporate reports, and wherever decisions about a population are made.

Background

Many distribution-based classification methods already exist and are tailored to locate and visualize particular classes of features. We will briefly describe some relevant related work from the past two decades and contrast how the methods described in the following chapters differ and extend existing approaches to feature classification.

Early work on volume rendering by Drebin et al. [12] assigns each possible density value in a CT data set a probability of belonging to a certain tissue class (e.g., air, fat, soft tissue, bone). Probabilities are assigned to the densities using the modes of a frequency histogram as frames of reference. To visualize a volume, then, each tissue class is assigned a unique color and each voxel is colored using the weighted sum of its tissue probabilities and the tissue colors. Levoy [43] provides an alternate concept of this so-called *transfer function*, the mapping from an observed intensity to color and opacity. Instead of predefined tissue classes, Levoy assigns non-zero opacity to a handful of contour values of interest. This approach in itself would yield visualizations of only thin surfaces of the contours, requiring expensive sampling rates to avoid aliasing. To ensure surfaces of some thickness, opacity is instead defined to be maximal for the target contour values but voxels having an intensity values near one of the target values are assigned a fraction of the maximal opacity. The fraction of opacity for these near contour values is inversely proportional to the magnitude of a voxel's gradient. In areas with faster rates of change within a volume, more voxels near the contour value are given partial opacity. The result is that contour surfaces have roughly constant thickness no matter how quickly intensities are changing. Levoy's technique can in a sense be considered as using a combination of scalar intensity and local neighborhood information to determine a voxel's classification. The limitations of these foundational works on volume rendering from the late 1980s is that they are designed for surfaces or predefined features in static scalar volumes. The techniques described in the following chapters readily support flexible feature specification in multivariate and temporal data.

In addition to serving as a useful standalone summary of a data set, histograms—discretized distribution functions—play a major role in guiding the user or an automated algorithm in setting parameters for transfer func-

tions. Besides directing the user in assigning opacity values of one-dimensional transfer functions [12, 46], histograms are integral in the semiautomatic generation of two-dimensional transfer functions described by Kindlmann et al. and Kniss et al. [37, 39]. In these works, appearing a decade after Drebin et al. [12] and Levoy [43], a scatterplot (a two-dimensional histogram, which records the joint distribution of two variables) of intensities versus gradient magnitudes is constructed, with prominent arcs marking the interfaces between materials. These arcs are used to assign non-zero opacity only to surfaces and zero opacity to homogeneous internal regions. Akiba et al. [2] extend histogram-based transfer function design still further by juxtaposing one-dimensional histograms for each timestep in a data set into a scatterplot. Users then use the scatterplot as a canvas for designing a temporal transfer function. The primary advantage of this approach is that the transfer function can change over time and be designed within the context of a volume's dynamic progression. All of these approaches are widely implemented and enable users to successfully visualize the classes of features for which they were developed. However, they all rely on features being based on scalar values and attributes derived from these values. As such, features are defined at a fairly low-level and the methods typically do not scale well to handle features defined across multiple variables. Additionally, the histograms on which these designs are based are all global histograms, in which low-frequency features may be obscured by higher-frequency phenomena.

Histograms have also been used extensively outside of transfer function design. Local histograms taken from the region surrounding a voxel have been employed to solve the partial volume effect, in which two or more materials overlap and classification is otherwise ambiguously determined [42]. Lundström et al. [44] describe a similar technique for disambiguating materials: the frequency at which a material occurs within its local neighborhood is combined with confidence intervals derived from domain knowledge to arbitrate between competing material classifications. In the same paper, they decompose an oversaturated global histogram into individual tissues by iteratively removing peaks according to their co-occurrence in the histograms of smaller regions. In another work, Lundström et al. [45] superimpose the histogram of a second variable within the confines of a traditional one-dimensional histogram. For

each bin of the first dimension, a histogram of an arbitrary second dimension is plotted using color between the x-axis and the first dimension's frequency on the y-axis. Gosink et al. [19] use scalar value distributions to automatically select an interesting contour value. On the resulting surface the correlation between two other variables is plotted. The number of diverse applications of histograms in the visualization process is immense, and allowing a general distribution querying framework is the goal of the work described in Chapters 1 and 2.

Navigating three-dimensional space to find features in data is not always straightforward, especially if visualization is not interactive. Data simulated or acquired in the twenty-first century especially has challenged the interactivity of our visualization software. In response, Bordoloi and Shen [7] define an algorithm for automatic viewpoint selection for volume rendering. Their algorithm evaluates the images for uniformly distributed viewpoints on the viewing sphere and scores them according to the images' entropies—or average information content—which is derived from what the authors call the voxels' visual probabilities and their user-defined importance. In Chapter 3, we apply a similar concept to find meaningful viewpoints for three-dimensional parallel coordinate plots. Instead of scoring a viewpoint according to entropy, we support a handful of metrics tailored for this visualization technique based on the distribution of depth and color values in the image plane.

Applying histograms to the visualization of time series data has also been discussed within the last several years, made possible by advanced storage and networking infrastructure, as well as a more widespread adoption of visualization in all areas of computational science. Wang et al. [63] profile the importance of a data block in a volume through time by examining its information content. The entropy of a block at one timestep is calculated using the probabilities derived from a multidimensional histogram of a block's feature vectors. Profiles are then clustered and the clusters are visualized. Younesy et al. [67] introduce a differential time-histogram table for accelerating visualization of time-varying data. Use this data structure, both spatial and temporal coherence are monitored between timesteps. Locations exhibiting a strong enough degree of coherence are not loaded from disk as the contour value or timestep changes. In both these works, the primary motivation for

using histograms is to accelerate analysis of unwieldy data sets. Histograms reduce complex data sets to a manageable summary that more easily fits in memory. The time histogram described in Chapter 4 serves a similar purpose for time series data that is specifically periodic in nature. The distribution query language of Chapters 1 and 2 is not aimed at accelerating processing and therefore does not discretize neighborhood distributions into histograms. Rather, the distributions are used to characterize features of interest in terms of their statistical moments.

Contribution

The following papers describe how we apply frequency information to three key arenas of visualization. In Chapters 1 and 2, we present a complete query visualization system called SeeDQ for 3- or 4-dimensional spatial data. To break beyond the limitations of using a single scalar value to isolate features, scientists express queries in SeeDQ based on the local neighborhood distribution for each data point. In Chapter 3, distribution analysis is used in image space to locate and display features in highly-multivariate and possibly non-spatial data sets rendered with parallel coordinates. Lastly, in Chapter 4, we describe a novel temporal histogram called a *periograph* for fast feature detection within periodic data sets. Both spatial and non-spatial data sets can be studied via periographs, which also serve as guide for formulating boolean range queries.

These tools provide a means for querying and visualizing new classes of features for all areas of data analysis. Using neighborhood distribution queries, users do not need to choose between simple scalar-based features and inflexible automated feature detection schemes. Distribution-based criteria can be powerfully and fluidly combined to locate and display domain-specific features. Parallel coordinate plots are a popular technique for displaying high-dimensional data, but they are prone to overplotting. Finding features typically has required some form of data reduction, but 3-D parallel coordinates avoids this step and guides users to viewpoint-dependent features particular to such displays. Lastly, extremely large periodic data sets from any field can be expressed very compactly in periographs, avoiding the need for users

to visually track features through time animations. To demonstrate their effectiveness, these tools are applied to data sets from the fields of medicine, physics, meteorology, and information visualization.

Chapter 1

Querying for Feature Extraction and Visualization in Climate Modeling

This chapter is a modified excerpt of a paper presented at the 2009 International Conference on Computational Science in Baton Rouge, Louisiana. It was published in Lecture Notes in Computer Science:

> C. Ryan Johnson, Markus Glatter, Wesley Kendall, Jian Huang, and Forrest Hoffman. Querying for feature extraction and visualization in climate modeling. In *ICCS 2009: Geo Computation*, volume 5545 of *Lecture Notes in Computer Science*, pages 416–425. Springer, 2009.

This paper is an abbreviation of a paper for which I served as the primary author and presenter. In terms of research, I was the primary contributor only to the work on neighborhood distribution queries, as described in Section 1.2. The sections of the original paper produced by my co-authors have been removed. The removed sections described a second query language which used a form of temporal querying based on regular expressions and a means of accelerated evaluation of boolean range queries. In Chapter 2, the work of Section 1.2 is expanded in greater detail.

Overview

The ultimate goal of data visualization is to clearly portray features relevant to the problem being studied. This goal can be realized only if users can effectively communicate to the visualization software what features are of interest. To this end, we describe in this paper a query language used by scientists to locate and visually emphasize relevant data in both space and time. This language offers descriptive feedback and interactive refinement of query parameters, which are essential in any framework supporting queries of arbitrary complexity. We apply this language to extract features of interest from climate model results and describe how they support rapid feature extraction from large datasets.

1.1 Introduction

Given the high-resolution and high-dimensionality of the datasets resulting from today's large-scale climate simulations, it is typically infeasible to visualize a dataset in its entirety. Moreover, the limits of hardware and human perception hinder real-time investigative analysis. Possible solutions to reduce the amount of visualized data may involve automatically detecting statistically unique locations [29] or using a problem solving environment supporting manual filtering of the data [34]. An alternative solution offering a balance between automation and control is to visualize or emphasize only a relevant subset of a dataset satisfying some query or hypothesis chosen by the user. To be effective, such queries must be expressed in terms of the problem domain and offer rich feedback, enabling interactive refinement of query parameters.

Modern fully-coupled general circulation models (GCMs) are frequently used to generate climate projections hundreds of years into the future. As spatial resolution and temporal output frequency increase, to better resolve and understand complex interacting phenomena, model output grows considerably. Analyzing this output through traditional means is becoming impossible. As a result, various data mining techniques, designed to extract features of interest and simplify very large time series datasets, are increasingly being applied to the climate modeling domain. Previous work by Hoffman *et al.* [24] has demonstrated the utility of such techniques by applying k-means cluster

analysis to hundreds of years of climate model results. This paper describes additional techniques that show promise in extracting regional and temporal features in an automated fashion.

Herein we describe a query language for visualizing features in any large, possibly time-variant datasets. The language was initially developed for volume visualization in a previous work [33] to enable users to express features in terms of statistical properties of local spatio-temporal neighborhoods. Querying is tightly integrated into the visual analysis process. We apply this framework to a recent climate modeling simulation.

1.2 Neighborhood Distribution Querying

Gaining insight into high-dimensional climate data often requires investigating and testing statistical hypotheses. For instance, scientists may be curious about the differences in the interannual variability of snow coverage between two decades, whether or not precipitation and temperature are positively correlated, or if mean solar radiation is decreasing through time. Investigating these hypotheses at a global scale reveals only overall trends. Generally, it is more informative to examine local regions independently and draw conclusions from observations of smaller-scale phenomena. To this end, we describe a framework that enables statistical querying and visualization of spatio-temporal data. We develop a visual query language in which scientists can express arbitrary, statistic-based queries to determine which local regions meet a set of hypotheses.

1.2.1 Neighborhood-based Querying

The core structure of our framework is the distribution of values in the local neighborhood around each data point. Scientists query for features of interest in terms of statistics derived from intervals within these distributions. Figure 1.1 illustrates the process of forming a query.

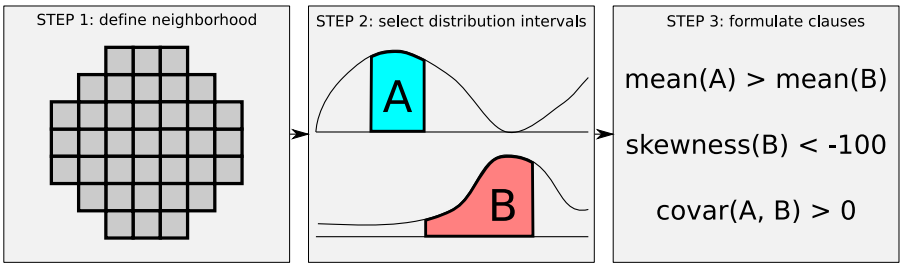


Figure 1.1: An illustration of the statistical querying process. (a) Scientists first choose the spatio-temporal neighborhood serving as the sample space. (b) Next, the sample space is refined to intervals of the variables and their distributions relevant to the problem. (c) Lastly, scientists describe features of interest using inequalities relating the intervals' statistical primitives.

Neighborhood

The exact size and shape of the local region that forms the sample space can be customized by the user. By default, the neighborhood is defined spatially as the set of data points within a specified Euclidean distance of radius r. However, users can alter the shape to search for features more easily described in an anisotropic sample space. Additionally, neighborhoods may be defined across both space and time domains.

Bin Selection

With the neighborhood defined, the user selects which variables are used to define the feature to be visualized. The user may focus on a particular interval of values for each variable. We refer to the interval on which a variable is examined as a *bin*. In choosing a bin, a scientist narrows the distribution for which statistical measures are calculated to a relevant subset of the data. For example, an investigation of liquid water may involve only temperatures above freezing.

Querying by Predicate Clauses

With the sample space and distributions in place, queries are now expressed as a series of inequalities relating properties of distribution intervals to target criteria. The properties currently supported include relative frequency, mean, variance and standard deviation, skewness, and covariance. As an ex-

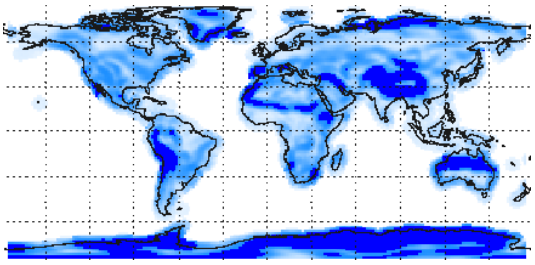


Figure 1.2: A query for surface temperatures in circular neighborhoods of radius 3 in July 2000 having a standard deviation greater than 3°K. The dark locations fully match the query, while the lighter shade's opacity reflects how nearly a non-matching neighborhood came to matching.

ample predicate clause, let us consider the feature of rapid surface temperature change in the spatial domain. We select a circular neighborhood of radius 3, a bin encompassing all possible surface temperatures, and a predicate clause stating that the standard deviation must be greater than 3. We apply this query to the July 2000 timestep of the IPCC land-model simulation, with the result shown in Figure 1.2. Large elevational gradients are the primary feature emphasized by this query because these are the neighborhoods with large temperature deviations.

In the proceeding example, the target criterion is constant: standard deviation must be greater than 3.0 for a neighborhood to match. However, we can also allow target criteria to be expressed dynamically in terms of other interval properties. To query for locations where the mean percentage of a land grid square covered in snow one decade is half that of another decade, we select a neighborhood spanning ten years, establish two intervals from 0% to 100% on snow coverage for each decade, and a clause requiring `mean(bin 1) < 0.5 * mean(bin 0)`.

Additionally, features typically involve compound criteria, and our query language readily handles multiple predicate clauses. Neighborhoods may match only a subset of the query's clauses, and in order to visualize partial matches, we record a *predicate signature*, which is a bitfield with bit i set to 1 if the criterion of clause i is fully met in a neighborhood. A neighborhood's predicate signature is used to determine its color when visualized. Each clause may also be assigned a weight reflecting its degree of importance in defining a feature.

To avoid a misleading boundary between matching and non-matching neigh-

borhoods, each neighborhood is also assigned a score in $[0, 1]$ indicating how closely the criteria of the clauses are met. The score for an individual clause is assigned according to an exponential decay function of a neighborhood's distance from the target criterion and a dropoff parameter that controls how quickly the score drops with distance, as detailed in Equation (1.2). A neighborhood's total score is defined in Equation (1.3) as a weighted sum of the clause scores.

$$\Delta = \begin{cases} 0 & \text{if criterion is met} \\ |lhs - rhs| & \text{otherwise} \end{cases} \tag{1.1}$$

$$score_i = \exp\left(\frac{\Delta^2 \times \ln(.5)}{dropoff_i^2}\right) \tag{1.2}$$

$$score = \sum weight_i \times score_i \tag{1.3}$$

1.2.2 Query Visualization

A query is visualized by coloring each location with its neighborhood's predicate signature, which represents the combination of clauses that are fully met. Scientists can interactively customize the colormap indexed by these predicate signatures or make a particular combination completely transparent. The neighborhood's score is used to increase the neighborhood's transparency further. By default, low-scoring neighborhoods are assigned a transparency close to 0, while high-scoring neighborhoods are more opaque.

Figure 1.3 is an example visualization of a query with multiple clauses. The query investigates the relationship between mean temperature increase and mean precipitation between the first and last decades of the IPCC dataset. Each color represents a different combination of the three criteria being met, though only six of the eight possible interactions actually occur.

1.2.3 Implementation

To formulate distribution queries for spatio-temporal neighborhoods, we have built a graphical tool that guides the user in selecting neighborhood sizes, bin ranges, and criteria. Changes to query parameters instantaneously update the

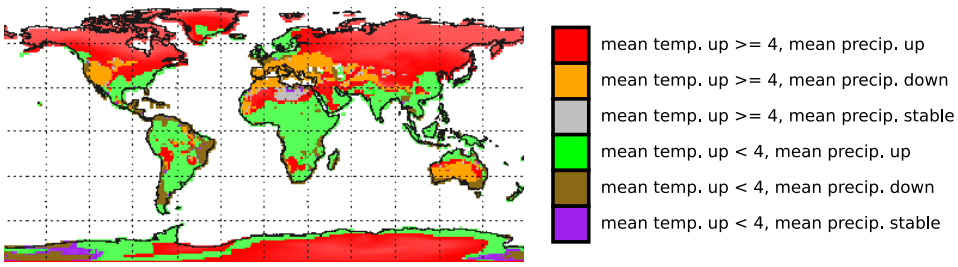


Figure 1.3: An investigation of mean temperature and precipitation between decades 2000–2009 and 2090–2099. Six possible interactions are observed in the simulation: mean temperature may or may not go up 4°K or more, and mean precipitation may increase, decrease, or stay constant. Each possible combination of events yields a unique predicate signature and color.

```
for each neighborhood h
   for each neighbor n
      for each variable v to be binned
         if n.v in bin
            add n.v to bin
   calculate bin statistics
   for each clause c
      if c.criteria met
         c.score = 1
         set bit in h.predicate_signature
      else
         calculate c.score according to dropoff
      h.score = h.score + c.weight * c.score
```

Figure 1.4: The algorithm for evaluating a neighborhood distribution query. The output of this algorithm is a set of predicate signatures and scores for each neighborhood in the dataset.

visualization for immediate and interactive feedback. Using the mouse, users can hover over each pixel and see exactly which clauses are fully met without having to decipher the colormap. A separate pane allows each individual clause to be studied in isolation from the entire query. The visually-composed neighborhood, bin, and clause information can be saved in an XML format and reloaded and modified as needed.

1.3 Conclusion

We have described a query language for extracting features from complex scientific datasets, and have demonstrated its utility by applying it to extract and visualize features of interest from climate model output. Such tools are becoming increasingly important as simulations generate higher spatial and temporal resolution datasets that necessitate use of novel techniques for analysis. Neighborhood-based querying is useful for extracting spatial patterns from large datasets based on the statistical parameters of a region's frequency distribution.

Chapter 2

Distribution-Driven Visualization of Volume Data

This paper is a reformatted work originally accepted for publication in a forthcoming issue of the journal *Transactions on Visualization and Computer Graphics*:

> C. Ryan Johnson and Jian Huang. Distribution driven visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):734–746, 2009.

I am primary contributor and author of this paper, having developed the software tools, generated the distribution queries, and written most of the text.

Overview

Feature detection and display are the essential goals of the visualization process. Most visualization software achieves these goals by mapping properties of sampled intensity values and their derivatives to color and opacity. In this work, we propose to explicitly study the local frequency distribution of intensity values in broader neighborhoods centered around each voxel. We have found frequency distributions to contain meaningful and quantitative information that is relevant for many kinds of feature queries. Our approach allows users to enter predicate-based hypotheses about relational patterns in local distributions and render visualizations that show how neighborhoods match

the predicates. Distributions are a familiar concept to non-expert users, and we have built a simple graphical user interface for forming and testing queries interactively. The query framework readily applies to arbitrary spatial datasets and supports queries on time variant and multifield data. Users can directly query for classes of features previously inaccessible in general feature detection tools. Using several well-known datasets, we show new quantitative features that enhance our understanding of familiar visualization results.

2.1 Introduction

A primary research goal of volume visualization is to develop methods that, for a given problem, can discriminate between relevant and irrelevant data. By isolating features of interest, users can more quickly make important scientific and medical decisions. To this end, visualization and feature detection have become inseparably linked in the task of understanding volume data. For thorough investigation, feature detection must be customizable, interactive, and performed in terms of the problem domain.

The visualization field has made a continuing effort to expand the types of physical properties that are considered in describing features. As a result, more classes of features can be visualized and examined. So far, the set of properties used has grown to include intensity, shape, curvature, orientation, size, and so on [43, 37, 38, 54, 5]. This collection of primitives allows users access to a rich set of features relevant for many applications.

Several researchers have taken steps toward defining features based on statistical information derived at each voxel. For example, Lundström et al. [44] detect features by considering material frequency within a neighborhood and Jänicke et al. [29] use a measure of local statistical complexity to automatically find probabilistically unique features. These works, discussed in greater detail later, have shown the power of defining features according to statistical data.

In this work, we incorporate the arbitrary use of statistical data into feature querying. Our key concept is to query for features in terms of the most general form of statistics—the probability distribution function. In particular, we incorporate dynamic, user-driven, and statistically-based feature queries that enable interactive investigation of volume data.

Describing features in terms of distribution functions is much more complex than typical scalar-based methods. This complexity becomes acute when multiple variables, and hence multiple distribution functions, are studied together to understand their combined effects. To address this challenge, we have developed a compact, expressive query language called SeeDQ (for "See Distribution Query"). In SeeDQ, users filter distributions into relevant value intervals and compose a series of boolean predicates that relate statistical properties of these intervals. These predicate clauses define the features of interest. Each neighborhood in the volume has its statistical properties computed and is scored according to the degree to which the clauses are satisfied. The resulting query volume is rendered so that users can see how neighborhoods matched.

We avoid making a binary classification of matching and non-matching neighborhoods; instead, the degree of match is computed as a scalar-valued score. This score serves as a fuzzy metric which can be used as input to a traditional transfer function. A second input is the neighborhood's type of match, which indicates which predicate clauses in a compound query were matched and which were not. Currently, SeeDQ modulates opacity using a neighborhood's score and assigns color according to the combination of matched clauses.

Our method enables users to specify in succinct terms the statistical frequency patterns to visualize. Patterns in the distribution correlate with many practical features of interest, and by using the distribution as the basis for feature identification, we allow classes of features previously inaccessible to be visualized, offering a novel and quantifiable perspective of the data.

Some examples of feature queries that are best defined using distributions include finding regions in which two materials are mixed in a certain proportion, filtering neighborhoods that maintain some degree of homogeneity between timesteps, or discovering how multiple medical scans relate to each other. In all these cases, sample-specific properties do not provide sufficient information to adequately answer the query, while spatial properties are sensitive to orientation changes.

Distribution queries yield descriptive visualizations that can be used to test quantitative hypotheses, but they can also be used to enhance our understanding of familiar features. For example, a query for the presence of an isovalue in a neighborhood can be combined with a query on covariance in order to

learn how variables interact around an isosurface. Alternatively, the output of a query can be used to augment existing feature detection mechanisms and create more powerful transfer functions and segmentation algorithms.

As a test of our distribution-based feature extraction method, we have been able to visualize previously inaccessible features that enhance our understandings of familiar results from several widely-used research datasets, including the 100-timestep Shockwave, 99-timestep Vortex, Visible Male head, and Chest datasets.

The remainder of the paper is organized as follows. Previous work related to our research is discussed in Section 2.2. In Section 2.3, we describe our concept of local distribution, and define the types of predicates that are supported. In Section 2.4, approaches to efficiently compute and render the resulting query volume are presented. Results and discussion follow in Section 2.5.

2.2 Related Work

Our method of multidimensional feature detection operates on the distribution of volume attributes in the neighborhood around a voxel. Feature queries are custom-tailored by the user using flexible boolean predicate-based criteria. This framework lends itself directly to comparing and visualizing multivariate data. Thus, the related work falls into four areas: (i) high-dimensional feature extraction, (ii) neighborhood-based visualization, (iii) query-driven visualization, (iv) multifield visualization.

2.2.1 High-dimensional Feature Extraction

In order for features to be detected and visualized, they must first be described in terms of the physical properties available. Density and gradient strength, for example, are often sufficient to characterize major boundary surfaces between bodily structures of living organisms [43]. Surfaces form a useful and perception-oriented class of features, but these may communicate artificial boundaries that do not necessarily exist in less-structured datasets [50].

Transfer functions have served as a primary means for interactively performing feature detection. The limits of one-dimensional transfer functions have long been known, and numerous multidimensional techniques have been

studied. Tzeng et al. [61] describe a training-based transfer function design that uses scalar value, gradient magnitude, 6-neighbor values, and spatial location to discriminate between features and non-features. Kindlmann et al. [38] consider local surface curvature to classify samples. Many techniques examine derivatives and their zero-crossings [39, 23] to isolate surfaces and oriented structures.

These projects illustrate the importance of the local domain in feature extraction. However, surface- and gradient-based transfer function designs only indirectly consider a sample's neighboring voxels. In contrast, our approach makes consideration of the surrounding region explicit and adds support for multivariate volumes. In this way, we accommodate visualizing different classes of features that are not necessarily surface-based.

Lundström et al. [45] use sorted histograms to perform two-dimensional classification. In their work, a traditional 1-D histogram along a chosen dimension is displayed, but each bin is sorted and colored along the frequency axis according to a second dimension. Users can alter optical properties for selected regions on the histogram. Possible second dimensions that can be used for sorting include frequency and variance of a value range within each voxel's local neighborhood.

Kniss et al. [40] augment volume rendering by visualizing the statistical risk of choosing a particular surface given multiple competing segmentations. The actual process of feature definition is not a part of their work. Instead, they use a sample's vector of segmentation probabilities to alter the appearance of uncertain boundaries. Our system allows for a form of uncertainty visualization using approximate matching to feature classes, but we do not retain specific information on samples' partial membership in each feature class.

Isosurfaces form another class of features, and the task of finding interesting isosurfaces can benefit from higher-dimensional information. For example, spectra of surface characteristics may be computed and displayed, guiding users to scalar values that yield an isosurface having a certain volume, surface area, or curvature [5, 51]. Tenginakai et al. [59] present a similar idea but use neighborhoods' statistical moments to determine salient isovalues. We support the use of statistical moments like variance and skewness in our distribution query to allow for more general feature detection.

2.2.2 Neighborhood-based Visualization

Using neighborhood data directly to augment input to a function has been discussed in several works. Laidlaw et al. [42] determine the mixtures of materials within a single voxel by sampling the voxel numerous times, forming a histogram of the results, and probabilistically choosing which materials best comprise the distribution. Their work aims at solving the partial volume effect and is dependent on an understanding of which materials are contained in the volume.

Image processing techniques that rely on neighborhood data have been employed for many applications. Fang et al. [16] integrate convolution and other neighborhood techniques in the rendering pipeline to sharpen, filter, and perform edge detection in volume data.

Lundström et al. [44] define the term *range weight* as the frequency at which a material occurs within a neighborhood. They use range weights for two applications: (a) decomposing the global histogram into its unknown constituent materials by using the neighborhoods' range weights to determine what intensity ranges combine in smaller regions and (b) competitive classification, in which a neighborhood's range weights for several known tissue types are combined with domain knowledge to determine which tissue type occurs at a sample. Our work is similar to this second application in our leveraging of statistics of local value intervals to perform feature detection. A primary difference is that Lundström et al. arbitrate between features by locating range weights within static confidence intervals derived from domain knowledge and combining the resulting confidence levels. In contrast, we provide a more general framework in which users define features by comparing range weight and other statistical primitives to expressions of arbitrary complexity. These expressions need not be static—they may include other statistics of the neighborhood.

2.2.3 Query-driven Visualization

The process of designing a transfer function can be likened to querying a database. A subtle difference between the two is that many lookup table-based transfer function designs ask the user to assign optical properties to the whole

dataspace, while query-based approaches allow the user to directly describe the values of returned records. Our distribution-based feature detection design fits a more query-based model.

An example of a query-based visualization application is VisDB [35], which displays the results of queries on large multivariate databases according to how relevant data items are to a query. The motivation behind this tool is that traditional queries are dependent on precisely known keys, an assumption that precludes a more general, approximate search. Users do not always know what they are looking for, and a less strict querying process is needed with strong visual feedback to indicate how close records are to fully meeting query predicates. VisDB is not applied to volume rendering directly, but is used to visualize the relevance of the results returned from querying of an arbitrary database.

Doleisch et al. [11] describe a feature querying framework that lets users make multidimensional range selections in a brushing interface. Multiple range selections can be combined in arbitrary ways using logical operations. Our method is similar, but we operate within the local neighborhood and add the ability for users to specify probabilistic criteria on the value ranges.

The Scout project [47] supports a form of scripted querying directly on the GPU in its shader framework. Users can form boolean predicates to filter out value intervals for further processing.

2.2.4 Multifield Visualization

Many transfer functions are multidimensional in the sense that they take scalar value plus derived attributes like gradient magnitude as input. But visualizing data that is already multidimensional in the acquisition stage poses a significant roadblock to clear understanding. Cai and Sakas [8] describe a taxonomy of visualization methods for intermixing multivariate or multiple volumes. Most techniques of displaying multivariate data involve combining noticeably different textures, color, and glyphs into a single visualization. Stompel et al. [57] use non-photorealistic techniques to help users decipher multivariate computational fluid dynamics and time-varying data. They note that most scientific visualization is by definition non-photorealistic in that variables like pressure, electron density, and temperature offer no photographic model.

Woodring and Shen [66] introduce a method to composite and visualize multiple volumes. To facilitate understanding of the resulting visualization, each intermediate composition is visualized in a volume tree. By navigating this tree, the users receive feedback at each step in the composition and can quickly locate errors. Examining suboperations in an expression is not unlike assessing individual predicates of a compound query, and we employ similar techniques to indicate which predicates match in our distribution-based query.

Jänicke et al. [29] use a measure called local statistical complexity to reduce multifield data down to a complexity field in which regions are marked according to their degree of information. The complexity field is then visualized. Their method of feature detection is automatic, with features defined according to the dataset but in an application-independent manner. Drawing from a similar motivation, we support detection of features defined on multiple variables. Our measure of feature importance, however, is dynamically computed and based on users' distribution-based feature queries.

Correlation fields have been demonstrated as another means of exploring multifield volumes. Sauber et al. [55] condense a field's variables at each location into a measure of similarity. All combinations of variables are examined hierarchically using a graph structure. Gosink et al. [19] project the correlation field between two variables on an isosurface, possibly defined on a third variable. Potentially interesting isosurfaces are detected using distribution functions.

2.3 Distribution-based Feature Detection with SeeDQ

SeeDQ is a query-based mechanism for extracting knowledge directly from local frequency distribution data. Designed as an expressive query language, SeeDQ is intended for exploring previously unknown relationships in the data, particularly those that require unified studies of multiple variables. To this end, SeeDQ expands the current vocabulary available for quantifying features in visualization. It can also be used for studying the statistical behavior of many familiar kinds of features, such as isosurfaces.

In SeeDQ, the local domain is defined as the neighborhood centered around

each individual voxel. Users select intervals in attribute space to filter out value ranges of interest in each neighborhood. For example, a user may be curious how intervals of high density, moderate temperature, and low pressure compare. Neighborhoods of interest are then described using a predicate-based query that describes features in terms of statistical metrics of the selected intervals. The query is evaluated by computing the statistical metrics for all neighborhoods and determining how closely they meet the target metrics. Matching neighborhoods are rendered according to their degree of match and combination of matched clauses.

2.3.1 Local Distributions

The key structure of SeeDQ queries is the local distribution of intensity values. We consider the distribution of each voxel's local neighborhood and examine its statistics.

Neighborhoods

We define a neighborhood H around voxel v as a set of voxels v_i surrounding v. H serves as a sample space, and each v_i is an elementary event in space H.

The simplest and default neighborhood is the set of all voxels within a Euclidean distance d of v. Some applications, however, may benefit from a sample space tailored to the features of interest. In SeeDQ, we allow users to define custom sample spaces as a collection of 3-D coordinates relative to the origin. These coordinate masks are flattened into a 1-dimensional array of voxel address offsets the neighborhood's central voxel. This design allows flexible handling of spaces of all radii, shapes, distance functions, and dimensionalities.

The shape and distance function used for a sample space are important application-dependent considerations. For example, spherical neighborhoods align well with normally distributed phenomena, while an ellipsoidal or even curved neighborhood may offer a more suitable sample space for features oriented in a known direction. When voxels are anisotropic, the neighborhood sizes may be adjusted according to the disparity among the differently-spaced axes. The mask used for a query can be selected at runtime in SeeDQ.

Every voxel serves as the origin of one neighborhood, and in constrast to the work of Lundström et al. [44], we allow neighborhoods to overlap. We do not fit neighborhoods to voxels at volume edges, however, where a full neighborhood cannot be formed. Partial neighborhoods contain smaller numbers of voxels. These reduced sample spaces are not easily compared to the full, internal neighborhoods and are omitted from the query.

Frequency Distribution

After a neighborhood structure is defined, we examine every neighborhoods' frequency distribution of values. When restricted to small neighborhoods (e.g., a radius of 1) such as those used to compute gradient with central differencing, distributions may not contain significant information. Alternatively, overly large neighborhoods may mask the regional cues that denote features. Accordingly, neighborhoods should be sized so that their distributions contain enough voxels to adequately evaluate the query.

Scalar metrics derived from the local distribution have been shown to discriminate between features [44, 42]. Herein, we allow the user to query directly on the distribution function, allowing for a more diverse set of features to be studied. The distribution offers the user a rich description of the underlying data for finding features. For example, in the distribution of a medical dataset, the frequency relationship between soft tissue and bone is immediately perceived and the relationship can be described quantitatively. Alternatively, neighborhoods that are homogeneous have very concentrated unimodal distributions, while heterogeneous neighborhoods have modeless distributions with high variance. A gap in a distribution indicates a hard boundary between materials, while a skewed distribution tells us that a value range is asymmetric about its mean. Many physical phenomena can be described using distribution primitives, and relationships between such phenomena can be described quantitatively using these primitives. This fact forms the foundation of our feature detection scheme.

Note that histograms, which are discretized distributions, have long played a role in 1-dimensional and 2-dimensional transfer function designs [37, 39]. Our use of distributions is different, however, in that our system uses distributions based on neighborhoods rather than the entire volume, and instead of

using the global histogram to influence the transfer function, users query for features directly on the local distributions. Also, as explained in Section 2.3.2, we do not convert local frequency distributions into conventional discretized histograms, but operate on the original data values to query a neighborhood distribution.

2.3.2 Predicate-Based Querying

There are two parts to a distribution query in SeeDQ: bins and predicate clauses. In specifying bins, users establish intervals of interest that are used to filter the scalar values' distributions for each neighborhood. A query is then composed by specifying a series of boolean predicates that statistically relate features of the neighborhood's distribution intervals. For example, a query may be formulated by specifying that neighborhoods contain twice as many high density values (e.g., bone) as low density values (e.g., soft tissue), or that pressure in one timestep should be comparable in mean value to pressure in another timestep.

Bin Specification

First, the user decides what value intervals in the volume are of interest for querying neighborhoods. The user restricts the domain of the distributions to intervals that are relevant to the features of interest and discards the rest. All standard inequalities are supported in defining intervals, and intervals for different bins may overlap.

To examine and compare multivariate datasets, the user specifies the variable over which a bin interval spans. For each interval within each neighborhood, we compute statistics like relative frequency, mean, variance, standard deviation, and skewness. We also support joint intervals that link two variables for each location in order to query covariance.

The bin is filled by traversing a neighborhood H and "dropping" each voxel in if its value falls within the interval. The result is a set of values, as described in Equation 2.1. As the neighborhood's voxels are processed, we calculate the bin's statistics in an online fashion. For example, relative frequency, mean, and variance for an arbitrary bin_i in neighborhood H are calculated in the

```
bin 0 {
   attribute 0
   range 0.9 to 1.0
}
bin 1 {
   attribute 1
   range < 0.5
}
bin 2 {
   attribute 4
   range >= 0.9
}
```

(a) Bins

```
clause 0 {
   freq(bin 0) > 50%
   weight 50%
   dropoff 0.1
}
clause 1 {
   var(bin 2) < var(bin 1)
   weight 25%
   dropoff 10
}
clause 2 {
   freq(bin 2) > 2 * freq(bin 1)
   weight 25%
   dropoff 10
}
```

(b) Clauses

Figure 2.1: A textual representation of an example query. Queries are comprised of (a) bins and (b) clauses. Here, the user is interested in three value intervals: $[0.9, 1.0]$ for variable 0, $[0, 0.5)$ for variable 1, and $[0.9, 1]$ for variable 4. The compound query specification in (b) contains three predicates. Weight values like 50% and 25% indicate the importance of each predicate to the overall query. The dropoff values of 0.1 and 10 are parameters used to allow approximate matches as described in Section 2.3.3.

conventional way in Equations 2.2-2.4, with the other statistics for standard deviation, skewness, and covariance computed according to their definitions.

$$bin_i = \{v : v \in H \text{and } min_i \leq v \leq max_i\} \tag{2.1}$$

$$freq(bin_i) = \frac{|bin_i|}{|H|} \tag{2.2}$$

$$mean(bin_i) = \frac{1}{|bin_i|} \times \sum_{v \in bin_i} v \tag{2.3}$$

$$var(bin_i) = \frac{1}{|bin_i|} \times \sum_{v \in bin_i} (v - mean(bin_i))^2 \tag{2.4}$$

The bin descriptions are formulated by the user using a tool described in Section 2.4.1, which produces an XML-based query file. We show a clearer and equivalent textual representation of several bin descriptions in Figure 2.1(a).

Clause Specification

Once the intervals of interest are specified, the user specifies a compound query about how the intervals should relate. This is done through a series of boolean predicates that compare bin variables to other quantities. An example query specification is shown in Figure 2.1(b).

Together with the bins specified in Figure 2.1(a), we use Figure 2.1(b) to illustrate how a query is evaluated. The simple predicate `freq(bin 0) > 50%` matches neighborhoods where a majority of the voxels fall in the interval specified by bin 0. To require that bin 0 not contain any voxels, the predicate becomes: `freq(bin 0) == 0`.

In addition to relating bin variables to constant values, users can refer to other bin variables. For example, the predicate `var(bin 2) < var(bin 1)` finds neighborhoods that have a more homogeneous subset of voxels in bin 2's interval than in bin 1's interval by comparing variances. The bin statistics variance and standard deviation are useful when neighborhoods contain a significant number of voxels and the dispersion of values is relevant to features of interest. For instance, variance is high in neighborhoods containing boundaries.

Arithmetic expressions are supported in each clause to allow more complex relationships between variables. A predicate of `freq(bin 2) > 2 * freq(bin 1)` finds neighborhoods that have over twice as many voxels in bin 2's interval than in bin 1's interval.

The right-hand side of the inequality can be any standard algebraic expression operating on constants and bin variables. It represents the target value of the clause, and the inequality operator indicates how the bin statistic on the left-hand side must relate for the neighborhood to match. If the inequality is satisfied, the neighborhood matches the predicate.

2.3.3 Approximate Matching

Displaying more neighborhoods than just those that strictly match every predicate often leads to more insight into the dataset. In this way, users receives feedback even for queries that return few neighborhoods. Additionally, users may be interested in neighborhoods that match certain clauses or match the

entire query to a certain degree. A binary cutoff between matching and non-matching neighborhoods does not allow users to specify vaguely determined queries, and the resulting visualization may contain distractingly sharp edges. To facilitate more flexible queries and improve feedback, we support two methods of displaying approximately matching neighborhoods: fuzzy scoring and predicate-signature classification.

Fuzzy Scoring

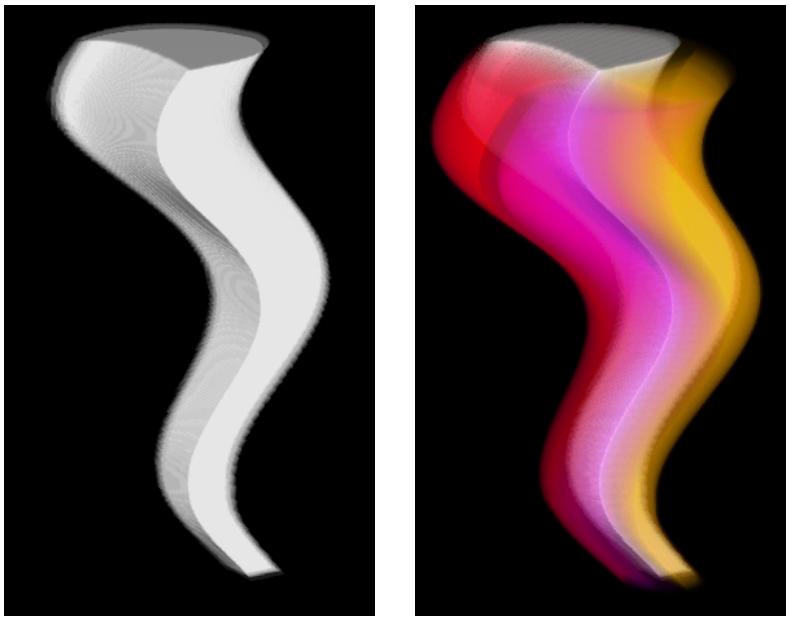
To illustrate our method of fuzzy scoring, consider the Tornado dataset [10] shown in Figure 2.2. Here, the user is searching for the intersection where the velocity magnitudes along all three axes are positive and in high concentration. This query can easily be formulated in terms of the frequency distribution. The rendering on the left shows the exact location of the intersection, but by allowing fuzzy matches to be shown with less opacity, the user also sees regions where the query is nearly met in the right image. In this case, the three adjacent, differently-colored areas are where only one or two of the velocity components are in high concentration.

To support fuzzy matching, we evaluate each predicate's inequality so that a value is returned indicating how far a neighborhood is from satisfying the inequality. This value is the predicate's *distance*, or Δ. When the inequality is completely satisfied, the distance is 0. Otherwise, we compute the distance as the absolute value of the difference between the left- and right-hand sides of the predicate's inequality.

For instance, using the first predicate from Figure 2.1(b), if `freq(bin 0)` does comprise over 50% of the neighborhood, it is assigned a distance Δ of 0. If only 40% of the neighborhood is in bin 0, then Δ is $|40\% - 50\%| = 0.1$.

The user chooses how loosely a query can be matched by specifying a dropoff parameter for each clause, as seen in Figure 2.1(b). The value of the parameter indicates the distance which should be assigned a score of 0.5. The dropoff parameter and Δ are then fed into an exponential decay function shown in Equation 2.5 to obtain the clause's matching score.

$$score_i = \exp\left(\frac{\Delta^2 \times \ln .5}{dropoff_i^2}\right) \tag{2.5}$$



(a) Non-fuzzy matching

(b) Fuzzy matching

```
bin 0 {
   attribute 0
   range > 0.01
}
bin 1 {
   attribute 1
   range > 0.01
}
bin 2 {
   attribute 2
   range > 0.01
}
```

(c) Bins

```
clause 0 {
   count(bin 0) >= 50%
   weight 33%
   dropoff 0.37
}
clause 1 {
   count(bin 1) >= 50%
   weight 33%
   dropoff 0.37
}
clause 2 {
   count(bin 2) >= 70%
   weight 33%
   dropoff 0.37
}
```

(d) Query specification

Figure 2.2: A single query on the Tornado dataset. Fuzzy matches rendered in (b) but not in (a). The opaque white intersection is comprised of neighborhoods where all three velocity components are positive and in high concentration. The three translucent regions in (b) consist of neighborhoods where only x- and z-velocities match, then y and z, and finally just z.

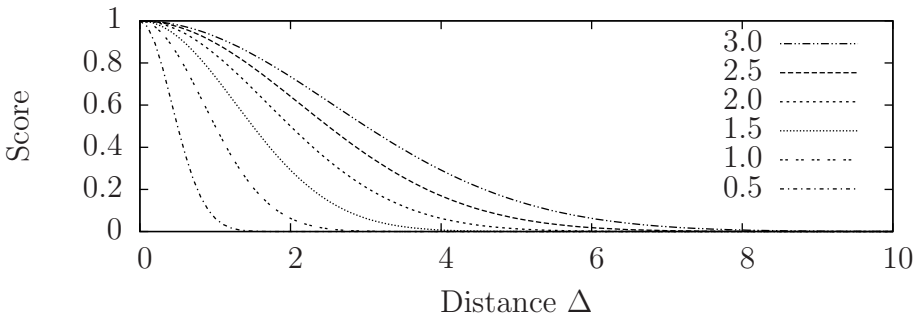


Figure 2.3: Several degree-of-match functions with differing dropoff parameters. The dropoff parameter indicates which distance is assigned a score of 0.5. Thus, a low dropoff will tolerate only closely matching neighborhoods, while a high dropoff offers a more approximate match.

We use the scoring function in Equation 2.5 rather than a linear ramp because it offers better control over scoring various degrees of non-matching. Additionally, this function is defined for any possible Δ, which is in $[0, \infty)$, and the function has a range of $[0, 1]$. This range is useful since the score values can then map directly to color or opacity during volume rendering. Larger dropoff parameters tolerate more loosely matching neighborhoods. Figure 2.3 shows how scores are computed for various dropoff values.

Both Δ and $dropoff_i$ are values in the same units as those returned by the bin statistic on the left-hand side of each predicate. If the predicate's statistic is `freq`, then the units represent a percentage of the neighborhood. If the statistic is `mean` or `stdev`, then the units are defined by the bin's attribute, and `var` is in square units. Normalization to a standard domain is avoided since distance is defined on an open-ended interval.

As in [35], we recognize that different predicates in the query may be of differing importance to the user. Therefore, in addition to a dropoff parameter, each clause is assigned a weight indicating its importance to the overall query, as shown in the Figure 2.1(b). This weight is used to directly modulate the clause's score and can be set to any positive or negative value. Negative weights serve as the negation operator in our matching scheme, used to penalize neighborhoods that match a clause. A weight of 0 is also legal; this indicates that no partial scoring is permitted and that the neighborhood must perfectly

satisfy the clause. In effect, a 0-weight restricts the query to regions where the specified condition is met, i.e., the other clauses become statements of conditional probability.

The score of the entire query is computed as the weighted sum of the scores of all clauses, as shown in Equation 2.6:

$$score = \text{clamp}(0, 1, \sum_{i \in Q} weight_i \times score_i) \tag{2.6}$$

Here, Q is the set of clauses forming the compound query. $score_i$ is the fuzzy matching score for clause i, defined by Equation 2.5, and $weight_i$ is the weight for clause i.

The weighted sum *score* is clamped between 0 and 1 to form a neighborhood's final score. The sum must be clamped because weights do not necessarily sum to unity, which allows users to execute different kinds of queries. For instance, a user may assign two predicates each a weight of 100% to look for neighborhoods that meet either predicate in a disjunctive relationship. If two predicates are each given a 50% weight, however, then both must be met in order for the final score to reach 1. This is a conjunctive query.

An alternative method of computing the neighborhood's final score is to avoid clamping and normalize the scores according to the sum of weights. However, this approach limits the types of clauses that can be expressed. The weights would have to sum to a non-zero value, i.e., two clauses with weights 1 and -1 would not be possible. Additionally, disjunctive clauses would not be supported, as a neighborhood matching one clause but not another would not receive a full score.

Obviously, combinations of weights should be tailored according to application needs. The final neighborhood score is saved with each voxel and can be used to modulate a sample's opacity in volume rendering. An example score volume can be seen in Figure 2.4(a).

Predicate-signature Classification

The neighborhood score gives a succinct measure of how closely a neighborhood matches the overall query, but it does not contain enough information to

determine how a compound query was matched. To allow for more in-depth feedback, we implement a system of *predicate signatures*. This technique is inspired by VisDB [35], in which a user can inversely determine which predicates of a compound query were matched by simply examining the results' color values.

The predicate signature of a neighborhood is a represented compactly by a bitfield for each voxel, with each bit corresponding to exactly one clause. Each clause whose inequality is fully satisfied (i.e., Δ is 0) has its bit turned on, and all remaining bits are turned off. When interpreted as an integer, the bitfield has a value between 0 and $2^{|Q|} - 1$, where Q is the set of clause predicates.

For typical queries, composed of 8 clauses or fewer, the bitfield fits comfortably in a single byte per voxel. Our approach readily accommodates larger bitfields, but the predicate signature is less useful when the number of predicates is high due to limits of human perception.

2.4 Query Volume Rendering

Users generate queries using an interactive graphical interface designed to give maximal feedback on changing query parameters. After the query has been adjusted to the user's liking, a query volume is generated and rendered using a GPU volume renderer.

2.4.1 Generating a Query

The ability to formulate arbitrary queries on statistical information means users can investigate any number of probabilistic hypotheses on their data. However, users often need guidance in selecting and iteratively refining query parameters, and, if neighborhood sizes are large, actually performing the query is quite computationally expensive. To guide users in setting parameters and to facilitate quick feedback, we have implemented an easy-to-use graphical tool for formulating queries and previewing their results.

Figure 2.5 contains a screenshot of our query formulation tool. The left pane contains GUI controls and histograms related to the query parameters. Users formulate a query by: 1) selecting neighborhood shape and size, 2) creating bins to filter out scalar ranges, and 3) creating clauses that relate the

bins' statistical metrics. The clause histograms show for all neighborhoods of the volume the distribution of the selected clause's statistical value and the distances between the actual neighborhood values and the inequality's target value. In the query depicted in the figure, distributions of the skewness of intensity range $[0, 200]$ and the distances from the target value of -1 are shown. Users can see how the score drops off with distance in the distance histogram.

The right pane renders a two-dimensional slice of the query volume. Both the currently selected clause and the total query are rendered. Users hover over a voxel with the mouse to see which queries it's neighborhood satisfies and how it is scored. They can interactively step through slices along any object-aligned axis and adjust query parameters for immediate feedback. When finished editing, users save the query to an XML file, which serves as input to a separate tool that queries the entire 3-D volume.

2.4.2 Generating Query Volume

To visualize a 3-D query volume, our search tool first loads the XML query file generated by the query formulation tool. The clause inequalities are lexically analyzed and parsed using structured grammars and code generated by the tools `flex` and `bison`. Since the inequalities contain neighborhood-specific terms that can only be evaluated after neighborhoods' bins have been filled, each boolean predicate is compiled into a postfix expression that can be evaluated quickly once its terms are known.

Each neighborhood can be processed independently of all other neighborhoods, so we have implemented our querying algorithm in parallel using MPICH. For a query on a cluster using n nodes, the input volume is first sliced into n subvolumes of approximately equal size. Slicing is done along the major axis for fast decomposition. To allow nodes to access all voxels in a neighborhood, some overlap of slices between nodes is necessary. Each node in the cluster processes all fully-contained neighborhoods in its subvolume, and the resulting scores and predicate fields are gathered back to the root node. This query volume is saved to disk for rendering.

2.4.3 Classification and Shading

The query volume is classified and shaded using a modified rendering algorithm. First, a sample's color is determined by its predicate signature. Essentially, the predicate signatures form a segmented or tagged volume. To determine a segment's color, we index into a colormap using the predicate signature. Since there are at most $2^{|Q|}$ possible signatures, the colormap contains a corresponding number of distinct colors. By default, the colors are selected from the outer ring of HSV hexcone at equidistant intervals, but these colors may be manually overridden. Effectively, this colormap translates each possible combination of matching clauses to a different color value, allowing the user to interpret how regions match the overall query.

A key issue of visualizing datasets of multiple variables is how to define the gradient at each sample to compute local illumination. Gradients are unlikely to correspond across variables. Furthermore, the composite volume has surface characteristics independent of the original variables, and it's unlikely that any combination of gradients could be used without introducing misleading surface cues.

We believe our query-based scoring approach leads to a convenient solution for determining the gradient. At each sample, we compute the central difference across neighboring scores. Scores gradate coherently from feature to feature, leading to well-defined and coherent surfaces around the matching neighborhoods. Since scores represent a weighted blend of the clauses of the user's query, we make gradients a meaningful user-driven attribute for volume rendering.

2.4.4 Visualizing Queries

We render the query volume using a GPU-accelerated volume renderer. View-aligned slices are clipped against the volume's bounding box and rendered, invoking a fragment program to determine each fragment's color using the scores and predicate signatures to index into the color and opacity textures.

Gradients are computed directly on the graphics card using the neighboring scores. Six additional texture lookups are required to compute the central difference at a sample, but the extra cost incurred by these lookups is minimal

given the limited number of texture channels available.

Tagged volume rendering poses several difficulties that we must address. Shading without the original data is overcome by using the scores in the query volume. By defining the gradient across scores instead of predicate signatures, smoother shading is achieved. Additionally, we must avoid linearly interpolating predicate signatures, since the interpolation may introduce incorrect intermediate values between segment boundaries. One solution [60] is to perform segment membership tests by performing interpolation for only one segment at a time. This has been implemented on the GPU [21] using a multipass scheme, with interpolation necessarily done without the aid of hardware acceleration.

Our solution to interpolating predicate signatures without sacrificing hardware-accelerated filtering requires loading the predicate signature (tag ID) volume twice, once with a nearest neighbor filter and once with a linear filter. To generate smooth color boundaries, we determine the predicate signature for each sample at subvoxel precision by examining three predicate signature values: the nearest neighbor t_n, the linearly interpolated value t_l, and the nearest neighbor t_g at the next voxel along the gradient. We select the sample's predicate signature t as the closer of t_n and t_g to the sample's interpolated t_l, as in Equation 2.7.

$$t = \begin{cases} t_n & \text{if } |t_l - t_n| < |t_l - t_g| \\ t_g & \text{otherwise} \end{cases} \tag{2.7}$$

Essentially, the linearly interpolated tag is used to determine whether the sample is closer tag-wise to the adjacent segment than the nearest segment. In homogeneous regions, $t = t_l = t_g = t_n$, so the tag is constant. We've found this method to greatly reduce the harsh edges of nearest neighbor interpolation without adding much complexity.

The interpolated predicate signature is used to index into the colormap. Each predicate signature is by default assigned a distinct color, which can be modified interactively. Gradients and opacity are defined from the score. Users can further modulate opacity based on the predicate signature, allowing for entire segments to be toggled on and off. Given the color, opacity, and gradient, the fragment's output color is computed.

Typically, the blending of transparent surfaces creates colors that map

Table 2.1: Resolutions and modalities of datasets used in example queries.

Dataset	Modality	Resolution		
		Variables	Spatial	Time
TSI	Simulation	5	$432 \times 432 \times 432$	300
Jet	Simulation	1	$256 \times 256 \times 256$	100
Tornado	Simulation	3	$256 \times 256 \times 256$	1
Head	CT	1	$128 \times 256 \times 256$	1
Nerve	Confocal Microscopy	1	$512 \times 512 \times 76$	1
Combustion	Simulation	5	$480 \times 720 \times 120$	122
Chest	CT	1	$384 \times 384 \times 240$	1
Vortex	Simulation	1	$128 \times 128 \times 128$	99
Climate	Simulation	71	256×128	1200

ambiguously back to their data sources. In practice, however, we have found that few enough clauses are used in SeeDQ queries that the predicate signature colors are sufficiently low in number to prevent any ambiguity.

2.5 Results and Discussion

We tested our method using a variety of medical and scientific datasets. The datasets used for the example queries are described in Table 2.1. For all queries shown in this work (excluding the temporarily-defined neighborhood in Section 2.5.4, we use spherical neighborhoods with a radius of 2 or 3 voxels.

2.5.1 Univariate Queries

In comparison to traditional feature detection using scalar values and 1-D transfer functions, our distribution query method can perform many similar tasks. In fact, SeeDQ can be considered a generalization of traditional preinterpolative 1-D transfer functions. To map a traditional transfer function to a SeeDQ query, we define a bin for each scalar value and add a clause that matches neighborhoods where the scalar value's frequency is non-zero. The predicate signature colormap is set so that the predicate signatures corresponding to each bin map to the color and opacity of the associated scalar value. Finally, we run the query using a neighborhood containing only one voxel.

Queries in SeeDQ also allow users to pick out scalar-based features that traditional transfer functions cannot. For example, Figure 2.7 shows where dense

bone structures occur in the Chest dataset. Using a 1-D transfer function, as in (a), small and thin bones cannot be distinguished from thicker bones since both have the same scalar value. Our distribution-based method, on the other hand, easily pulls out neighborhoods that contain a larger percentage of bone values. In (b), the fully matching neighborhoods are an opaque yellow, and the partial matches (i.e., the thinner bones) are rendered with limited opacity in cyan. A spherical neighborhood of radius 3 (123 voxels) was used as the sample space.

Alternatively, queries may consider the dispersion of values in neighborhoods. For instance, queries can examine a bin's variance metric to find regions of certain homogeneity. Figure 2.8 shows a query for extremely heterogeneous regions, as shown in white and blue in (b). Since the interface between the supernova and the zero-valued background surrounding it also has a high variance that normally occludes internal structures, we use a compound query to distinguish neighborhoods that do not contain background voxels. The resulting predicate signatures allow us to further discriminate the partially matched neighborhoods so that we can map the occluding surface to a minimal opacity for context.

SeeDQ queries enable users to perform custom filtering on volume data. Figure 2.9 shows one such filter on a nerve dataset containing a great deal of noise. The query targets neighborhoods that contain more foreground voxels than background. This is not unlike an erode operation in image processing, which could be used to locate weak structures. This query demonstrates a bin-to-bin comparison, allowing the target criteria to be defined in terms of the particular neighborhood being scored.

We've additionally found it useful to support predicates that must be mandatorily matched. Figure 2.10 shows an example of this, restricting querying to neighborhoods that contain at least 1% bone by specifying a weight of 0%. Another predicate is used to find neighborhoods that contain at least 20% air. In combination, the query finds bone in close proximity to air, revealing the sinuses and ear canals.

2.5.2 Multivariate Queries

Distribution queries easily extend to find features of arbitrary dimension. Any number of physical properties or timesteps can be considered by simply creating a bin to track the desired attribute for each neighborhood.

Figure 2.4 shows a query on two timesteps of the Jet dataset. Here, the user is interested in all scalar values greater than 200 and creates two bins, with bin 0 holding values from timestep 80 and bin 1 from timestep 99. The query states that for a full match, both bins must hold at least 80% of the neighborhood. In (a), the score volume is rendered using the score as a scalar value. Only along the central core of the shockwave do the scores reach 1.0. Outside of this core, the scores drop to 0 as neighborhoods contain fewer voxels having values greater than 200. In (b), we see how the two timesteps intertwine by coloring according to the predicate signatures. Cyan indicates that only timestep 80 was fully matched, magenta indicates timestep 99, and yellow indicates neither was fully matched. (The core where both timesteps match is occluded.) This type of query gives an indication of how much and where two timesteps overlap.

An example of using covariance to compare two timesteps of the Vortex dataset is shown in Figure 2.6. Here, one predicate is a query for neighborhoods where the two timesteps are positively correlated (orange), while another is a query for a negative correlation (purple). Two interesting structures appear in the rendered volume where a region of positive correlation envelops a region of negative correlation.

2.5.3 Case Study: Combustion Dataset

Even if a feature like an isosurface is familiar to users, one might still have questions regarding how the feature interacts with other variables. Distribution queries can be used to enhance the feature by incorporating further clauses. Here, we show a relevant scientific example from a study of combustion.

Figure 2.11 shows several renderings of the Combustion dataset from the Visualization 2008 Meet the Scientists panel. Combustion scientists are curious how regions of flame extinction behave through time [3]. It is already

known that flame extinction occurs in locations where the following criteria are observed: (a) a stochiometric mixture rate of 0.42, indicating the presence of flame, and (b) low oxygen content. However, how quickly these regions change in size and how they reignite in relation to other variables is not fully understood. To verify hypotheses made by combustion scientists, we apply SeeDQ in order to see how vorticity, the measure of rotational flow, varies with the amount of available oxygen through time. Four sample queries relating these values with different inequalities are shown in (c) through (e). The clauses used to generate these queries score neighborhoods according to how much of the correct mixture fraction and OH levels they contain and how mean vorticity and OH levels relate between timesteps 121 and 122, the final timesteps of the simulation. Neighborhoods surrounding the dissipating flame and having vorticity and oxygen increase or decrease in mean value in a certain combination are shown in gold, with partially matching neighborhoods rendered in green. In this way, scientists can use frequency queries to posit meaningful relationships between variables through time.

2.5.4 Case Study: Climate Dataset

We have also applied neighborhood distribution queries to a climatology simulation. The dataset consists of 71 variables measured as monthly averages for every month in years 2000 through 2099. Variables are measured across a discrete sampling of the Earth's surface with a grid resolution of 256×128. The simulation uses a land-only model; variables at not measured for bodies of water.

Our collaborators are curious how temperature and precipitation change through time across the Earth's surface. In this case, the neighborhood is defined to be temporal and not spatial, so the distributions we examine are comprised of data values for a single latitude and longitude coordinate through time.

Since the climatologists are curious about changes in two variables, we create four bins for our query: one for precipitation in 2000-2009, one for surface temperature in 2000-2009, one for precipitation in 2090-2099, and one for surface temperature in 2090-2099.

To determine how temperature and precipitation relate, we formulate clauses

to query for each possible change to the variables. Mean temperature may go up by some amount from the beginning of the century to the end, or it may go up by some smaller amount. (In this simulation, in no locations does mean temperature go down.) Mean precipitation may go up, stay the same, or go down.

Figure 2.12 shows the result of applying this query to the IPCC climate dataset. Our collaborators have verified the consistency of this result with their expectations. Extreme latitudes see more significant temperature increases, and relative precipitation levels generally are exaggerated—dry locations become drier and wet locations become wetter—as time progresses.

2.5.5 Timing Results

To distribute the computational burden of evaluating an arbitrary query on neighborhoods centered on all of a volume's voxels, SeeDQ operates in parallel on subvolumes of the original dataset. We tested our tools on a 16-node Linux cluster, equipped with dual-core 3.2 GHz Intel Xeon processors with 4 GB RAM. Given the natural parallelism of our approach, performance in our MPICH implementation scales linearly with the number of processors for a query containing three predicates and three bins, as seen in Table 2.2.

Processing times are not interactive due to the high computational demand, which is proportional to the neighborhood size, the number of bins and clauses, and the number of full neighborhoods contained in the volume. For this reason, we recommend that queries first be built using the query formulation tool, which operates at interactive speeds on a single preview slice of the volume. After testing and tweaking, the parallel 3-D query tool should be used to produce full volume renderings.

A considerable amount of overhead is introduced by optimizations, but the benefits are quickly seen as processing times grow by a much smaller factor than do the neighborhood sizes. Given that small neighborhoods have less statistical significance, we favor performance for larger neighborhood sizes.

The GPU-based volume renderer was built using the OpenGL Shading Language (GLSL) and operates much like traditional hardware-accelerated slice-based volume renderers. Framerates on an NVIDIA GeForce 8800GT are approximately 25 fps for a 256^3 volume, 512 by 512-pixel viewport, and a

Table 2.2: Average processing time per node for a three-bin, three-predicate query on the 256^3 Tornado dataset for spherical neighborhoods of increasing size and voxel count. In this example, the number of voxels in each neighborhood is a count of all voxels within a Euclidean distance of the radius value.

Neighborhood Size		Mean Query Time Per CPU (seconds)						
		1 Node	2 Nodes		4 Nodes		16 Nodes	
Radius	Number of Voxels	Time	Time	Speedup	Time	Speedup	Time	Speedup
1	7	15.70	6.06	2.6x	3.41	4.6x	0.92	17.1x
2	33	19.98	9.86	2.0x	4.29	4.6x	1.28	15.6x
3	123	33.08	12.18	2.7x	7.16	4.6x	2.04	16.2x
4	257	52.08	26.18	2.0x	13.30	3.9x	3.24	16.1x

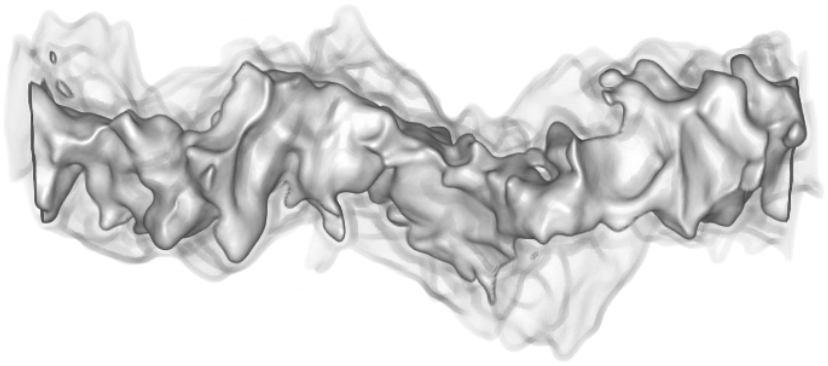
sampling rate of 1 sample per voxel. The added complexity of interpolating predicate signatures at subvoxel precision is minor compared to a multipass solution to this problem, incurring a cost of 5.5 ms per frame compared to a reference volume renderer.

2.6 Conclusion

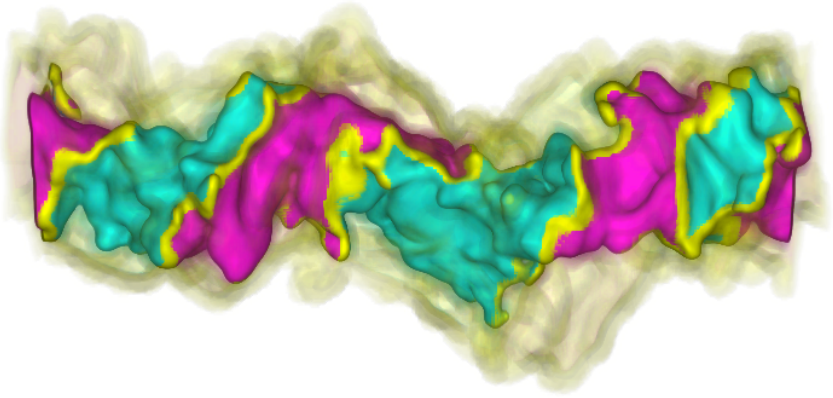
Local frequency distributions offer users a new handle on gaining insight from volumetric datasets. By querying for features in distribution space with SeeDQ, we can see new results that agree with and enhance traditional feature detection methods based on scalars and gradients, in addition to locating a whole new class of features that were not previously accessible to or easily specified by the user. Furthermore, features may be defined across any number of variables or timesteps. The resulting query volume can be rendered for real-time feature analysis. Features can be quantitatively described in two ways from the generated imagery: 1) by associating color and the corresponding set of matched clauses, users can describe what combination of criteria a neighborhood meets; and 2) by using the preview tool to examine exactly how and to what extent a location matches both individual clauses and the entire query. Such meaningful interpretation and feedback is essential to the process of investigating our data.

For future work, we are interested in supporting more distribution primitives for detecting other kinds of features. We also want to investigate building

into our query formulation tool mechanisms so that users can better investigate neighborhood distributions and how they correlate to features. Distributions are generally observed on a volume-wide scale, and the patterns seen in smaller regions may not be widely known. The creation of custom neighborhood shapes and selection of neighborhood size could be enhanced by allowing users to select a prototype neighborhood containing a known feature. Additionally, it would be interesting to adjust neighborhood shapes and sizes according to their context. This flexibility would accommodate features that vary in size. Finally, we believe a GPU-implementation of SeeDQ would realize significant reductions in computation time.



(a) Score Volume



(b) Query Volume

```
bin 0 {
   attribute 0
   range > 200
}
bin 1 {
   attribute 1
   range > 200
}
```

(c) Bins

```
clause 0 {
   freq(bin 0) >= 80%
   weight 50%
   dropoff 0.83
}
clause 1 {
   freq(bin 1) >= 80%
   weight 50%
   dropoff 0.83
}
```

(d) Clauses

Figure 2.4: A query on a volume containing timesteps 80 and 99 of the Jet dataset. The grayscale score volume in (a) is opaque where the query is fully met, i.e., both timesteps have 80% of their values in the correct interval. Elsewhere, scores dropoff to 0. In (b), the entire query volume is additionally colored according to its predicate signatures to indicate which clauses were matched.

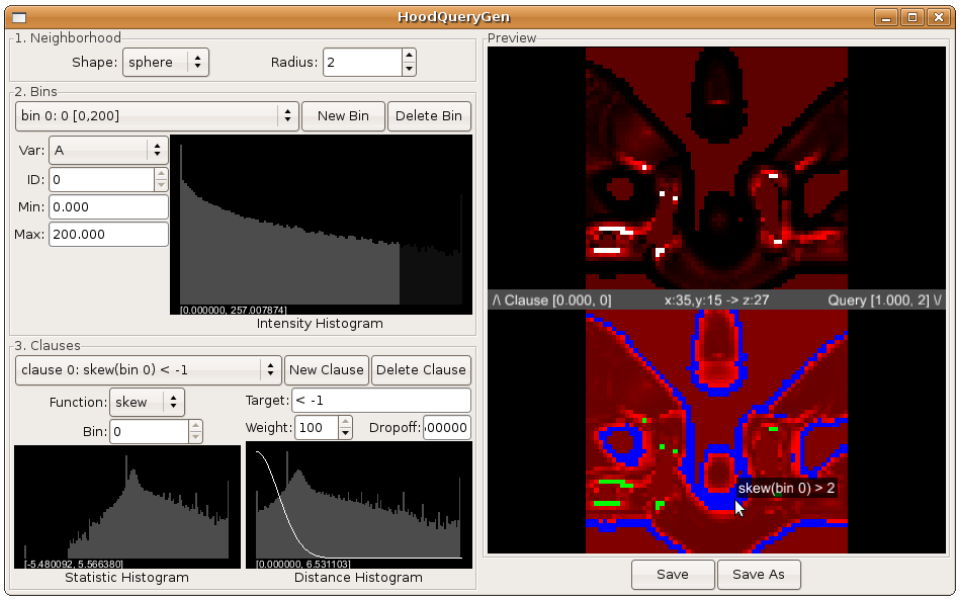
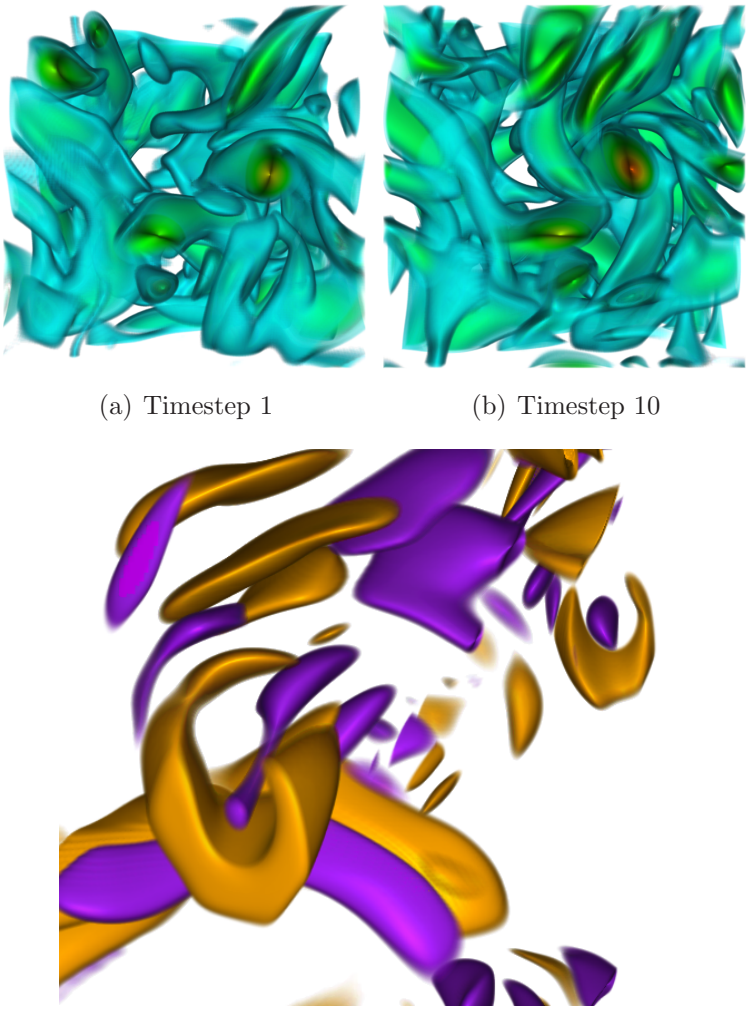


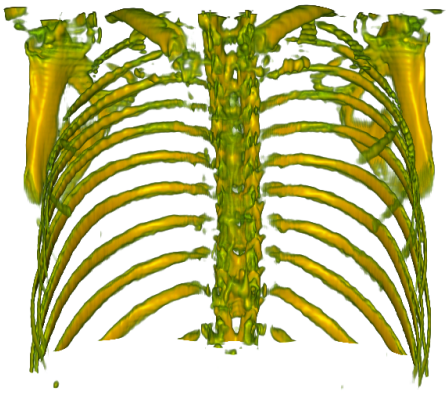
Figure 2.5: A screenshot of our tool for quickly generating neighborhood queries. Controls for setting neighborhood shape, creating bins, and creating clauses appear in the left pane. A single slice of the queried volume is rendered in the right pane. The slice is scored and rendered according to both the currently selected clause (top) and the entire query (bottom) for maximum clarity of how the query is being matched. Color indicates which combination of clauses matched and intensity indicates score. The color mapping is described further in Section 2.4.3. The query shown focuses on the interval $[0, 200]$ of the Neghip dataset and contains two clauses, one highlighting neighborhoods of positive skewness (blue) and the other negative (green).



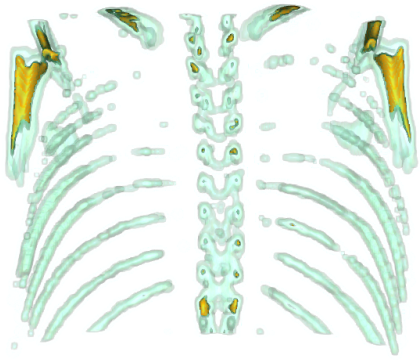
(a) Timestep 1 (b) Timestep 10

(c) Query Volume

Figure 2.6: A studying of timestep correlation in the Vortex data. (a) and (b) Timesteps 1 and 10, respectively, of the Vortex dataset, rendered with a 1-D transfer function. (c) A query on a volume containing the two timesteps in (a) and (b). Here, the query is disjunctive, requesting neighborhoods with either positive or negative covariance between the two timesteps. Orange indicates that the two timesteps have a direct correlation, while purple indicates that they are indirectly correlated.



(a) Scalar Volume



(b) Query Volume

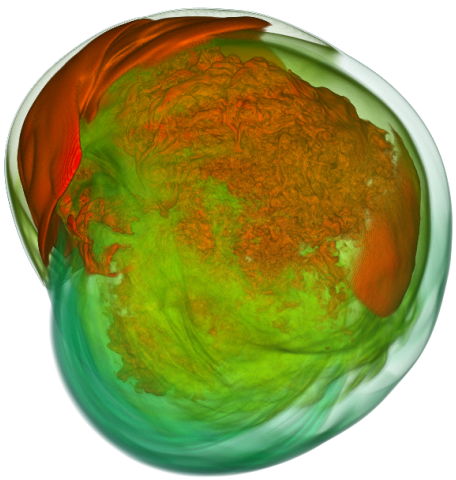
```
bin 0 {
   attribute 0
   range > 210
}
```

(c) Bins

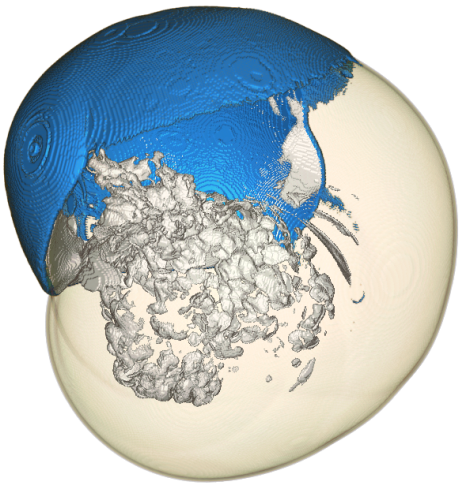
```
clause 0 {
   freq(bin 0) >= 70%
   weight 100%
   dropoff 0.53
}
```

(d) Clauses

Figure 2.7: A query for large bone structures in the Chest dataset. (a) A 1-D transfer function can only distinguish between bone and not bone. (b) Our distribution-based query finds large bones in the spine and shoulder. Smaller bones have lower scores and are shown with limited opacity. A spherical neighborhood of radius 3 was used as the sample space.



(a) Scalar Volume



(b) Query

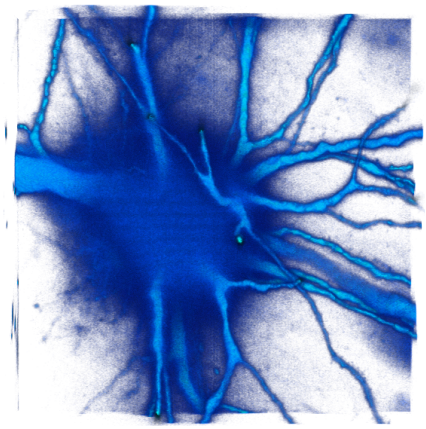
```
bin 0 {
   attribute 0
   range < 30
}
bin 1 {
   attribute 0
   range > 50
}
```

(c) Bins

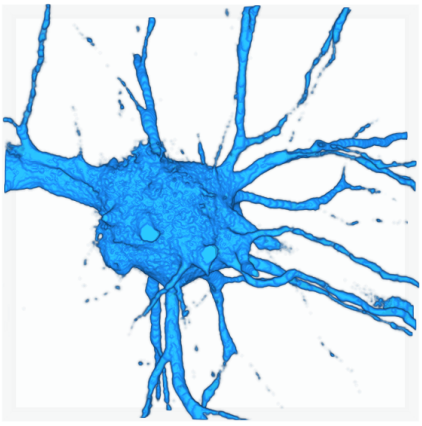
```
clause 0 {
   freq(bin 0) == 0%
   weight 10%
   dropoff 1.18
}
clause 1 {
   var(bin 1) > 1000
   weight 90%
   dropoff 5.27
}
```

(d) Clauses

Figure 2.8: A neighborhood query of the TSI supernova simulation. (a) Entropy field for a timestep of the simulation, rendered using a 1-D transfer function. (b) A query on the same timestep for heterogeneous regions with and without zero-valued background voxels. Blue and white regions have high variance, transparent red has a somewhat high variance, and blue and transparent red contain some amount of background.



(a) Scalar Volume



(b) Query Volume

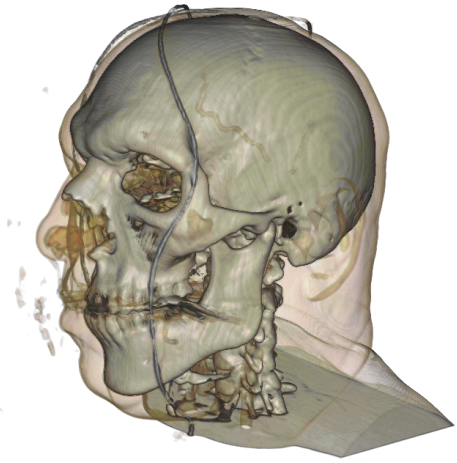
```
bin 0 {
   attribute 0
   range 0 to 39
}
bin 1 {
   attribute 0
   range 40 to 255
}
```

(c) Bins

```
clause 0 {
   freq(bin 1) > freq(bin 0)
   weight 100%
   dropoff 0.37
}
```

(d) Clauses

Figure 2.9: Neighborhood queries as a means of filtering. (a) Traditional volume rendering of a noisy nerve dataset. (b) A distribution query for neighborhoods containing more non-zero voxels than background. SeeDQ allows users to perform custom filtering using neighborhood distribution criteria.



(a) Scalar Volume

(b) Query

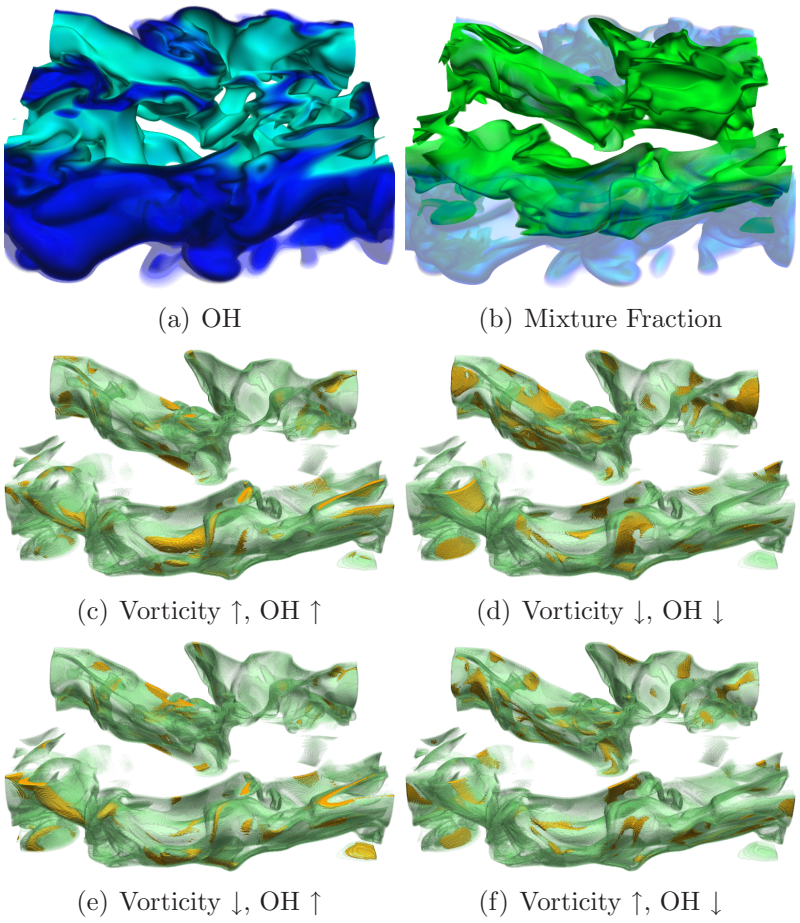
```
bin 0 {
   attribute 0
   range 0 to 24
}
bin 1 {
   attribute 0
   range > 100
}
```

(c) Bins

```
clause 0 {
   freq(bin 1) > 1%
   weight 0%
}
clause 1 {
   freq(bin 0) > 20%
   weight 100%
   dropoff 0.12
}
```

(d) Clauses

Figure 2.10: Neighborhood query on Visible Male dataset. (a) Volume rendering of the Visible Male dataset using a 2-D transfer function. (b) A query for neighborhoods that must contain bone and some amount of air. Those with a lot of air, like the sinuses and ear canal, are rendered in magenta. Those with less air are rendered mostly transparent for context. A weight of 0% forces the associated clause to be met for any non-zero score to be assigned.



(a) OH (b) Mixture Fraction

(c) Vorticity ↑, OH ↑ (d) Vorticity ↓, OH ↓

(e) Vorticity ↓, OH ↑ (f) Vorticity ↑, OH ↓

Figure 2.11: Querying for temporal and multivariate features in a combustion simulation. (a) Traditional volume rendering of low OH content from timestep 122 of the simulation. A rainbox colormap is used, with value $7 \cdot 10^{-4}$ colored blue-green. (b) Traditional volume rendering of mixing rate less than 0.42. Value 0.42 maps to green. (c)-(e) Several queries on vorticity and oxygen content through time around locations of flame dissipation. Flame dissipates where OH $< 7 \cdot 10^{-4}$ and the mixture fraction equals 0.42. Neighborhoods containing these isoranges and having mean vorticity and oxygen content increase or decrease in a certain way through time are shown in gold. For instance, (c) is a query for areas of dissipation where vorticity and oxygen are both increasing in value from timestep 121. Neighborhoods not completely meeting the dissipation ranges or not satisfying the vorticity-oxygen clauses are shown in green. (d) Vorticity and OH both decreasing. (e) Vorticity decreasing, OH increasing. (f) Vorticity increasing, OH decreasing.

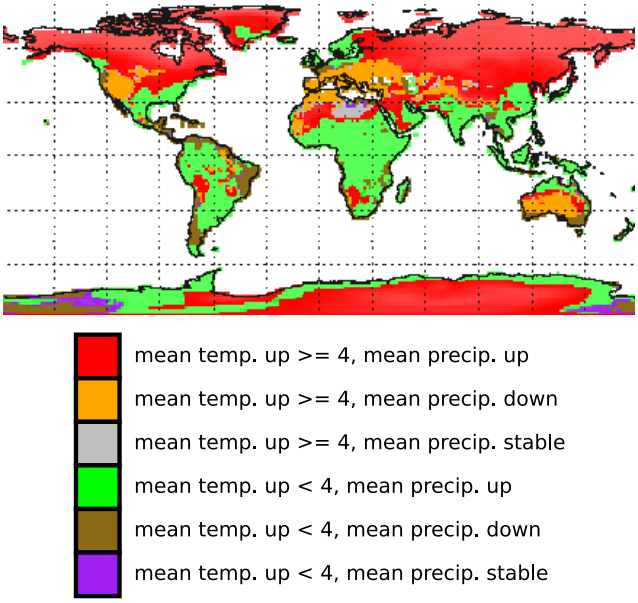


Figure 2.12: A query on the interrelation between precipitation and temperature through time on a climate simulation. The color legend indicates how the means of the two variables move in tandem from the first decade of the millenium to the last decade. Clauses query for locations where temperature increases by more or less than 4 degrees Celsius and where the amount of precipitation increases, decreases, or holds constant.

Chapter 3

Unoccluded 3-D Parallel Coordinate Display

This paper is currently under revision and is to be resubmitted in the near future to an information visualization conference. I am the primary contributor and author of this work.

Overview

Parallel coordinate plots are a primary means of exploring high-dimensional data and detecting variable relationships. However, plots quickly become cluttered as datasets grow in size, and variable relationships become indiscernible. Methods of clustering and filtering have been studied to alleviate occlusion, but most such methods involve data reduction. We propose a method of 3-D parallel coordinate display that requires no data reduction and yet facilitates easy detection of correlations between variables. Each coordinate line is given a depth value proportional to one of its data values, and the collection of lines is rendered as a 3-D shaded surface. Relationships between variables are observed by interpreting the lighting cues between axes. Additionally, we discuss a method of automatic feature detection using optimal viewpoint calculation designed especially for 3-D parallel coordinate plots.

3.1 Introduction

Parallel coordinate plots (PCPs) are a visualization technique for examining relationships in multivariate data. In PCPs, each n-dimensional observation in a dataset is plotted as a polyline connecting locations on a set of n parallel axes spanning the dimensions' value spaces. By adjusting a colormap and investigating spatial relationships of the lines, users leverage the human perceptual system to detect trends and correlations between variables.

As the number of observations within a dataset grows, coordinate lines are increasingly likely to intersect in the plot. As the density of intersections increases, lines are obscured and the user typically must manipulate filters to reveal hidden portions of the dataset. Interacting with filter controls is often not a straightforward process, and important features may remain hidden or be unintentionally removed. Alternatively, lines may be clustered and coalesced so that overdraw is minimized. Either solution involves altering or reducing the original data, which may, in turn, hide features from the user.

We present a new technique of displaying parallel coordinates in 3-D that addresses the problem of occlusion without eliminating data. Instead of drawing all lines in a single plane, we give lines a third spatial dimension proportional to a selected variable. The user interactively rotates the PCP to see occluded features. Additionally, since we plot lines in 3-D, we are able to shade lines to better emphasize trends in locations of dense plotting, where the lines appear as connected surfaces.

Extending PCPs to 3-D introduces the problem of assuring that the plot is viewed from a location that reveals important features. We address this problem by detailing a mechanism for calculating optimal views. Different classes of features may be of interest to the user, so we define several criteria used to quickly locate meaningful viewpoints.

The remainder of this paper is organized as follows. We discuss work relevant to 3-D PCPs in the next section. In Section 3.3 we describe how we extend PCPs to 3-D. Our method of calculating optimal viewpoints for several classes of features is detailed in Section 3.4, followed by a discussion of our method in Section 3.5. Finally, we conclude and suggest future directions for our work.

3.2 Related Work

Parallel coordinates are a popular means of displaying and interactively investigating high-dimensional data [28, 27]. The problem of occlusion in PCPs has been studied extensively. Most solutions involve either data reduction to prevent or diminish occlusion or alternate visual arrangements that are less prone to overdraw.

3.2.1 Data Reduction

Sampling is a straightforward approach of reducing clutter in PCPs by simply eliminating lines from the display. Ellis and Dix [14] develop criteria to measure occlusion so that the sampling can keep overdraw below some tolerable level. Since eliminating data may mislead viewers, they sample only within a draggable sampling lens.

Another popular means of reducing the data is to cluster lines and draw representations of each cluster instead of individual lines [17, 48], perhaps preserving outliers [49]. Some researchers do not eliminate lines, but draw lines in clusters of low interest with limited opacity [31, 4]. Our approach does not cluster the data, but clustering may be performed as a further means to reduce occlusion.

3.2.2 Visual Arrangement

To minimize occlusion without eliminating or diminishing data, parallel coordinate lines may be displayed by an alternative means. Some approaches retain a traditional 2-D display, but several projects have extended into 3-D.

2-D Arrangement

One way to reduce occlusion without modifying the underlying data is to reorder the axes. Peng et al. [52] define a clutter metric for an arbitrary axis ordering as the average number of pairwise outliers between adjacent axes. All possible orderings are examined, and the ordering with the lowest clutter metric is selected as optimal.

Shading and geometry have been used to make features more apparent. For example, Graham and Kennedy [20] and Zhou et al. [68] replace each polyline with a curve to make perceiving an observation's continuity across each axis easier. McDonnell and Mueller [48] extend this technique to clustered displays and incorporate halos and shadows for depth cues.

3-D Arrangement

Parallel coordinate displays have also been extended into 3-D in a variety of ways to overcome the limits of traditional 2-D plots. Fanea et al. [15] "fan" out a PCP by rotating each successive coordinate line around an axis connecting the bases of the parallel axes. Each parallel axis effectively is transformed to a star glyph, allowing viewers to see inside a dense dataset. Johansson et al. [30] present a display that allows users to select a focus variable and view its relationships to multiple other variables. In traditional 2-D plots, each variable is directly connected to at most two neighbors, but their approach distributes an arbitrary number of neighboring axes in 3-D around the focus axis.

An approach similar to ours is applied to ThemeRiver [22] visualizations by Imrich et al. [26]. ThemeRiver is a method of displaying time-dependent categorical data as flows of color that widen and narrow according to a category's change of prevalence through time. Imrich et al. project these flows onto a 3-D Bézier surface, in which the surface height represents the magnitude of a second variable, in order to depict added information in an otherwise 2-D display. Since PCPs already are able to compactly present high-dimensional data, our application of this technique is different: we apply 3-D extrusion for the purpose of clutter reduction.

Several researchers have explored 3-D PCPs in which each coordinate line is given a third dimension. Wegenkittl et al. [65] assign depth according to a line's discrete timestep. Since many datasets exhibit temporal coherence, adjacent data points are connected to produce a smooth surface representing the dataset's evolution through time and variable space. They generalize this approach to produce a trajectory manifold with arbitrarily-oriented interaxis planes. Our work also extrudes and shades parallel coordinate lines, but our goal is to maintain traditional PCP interaction and interpretation. Honda

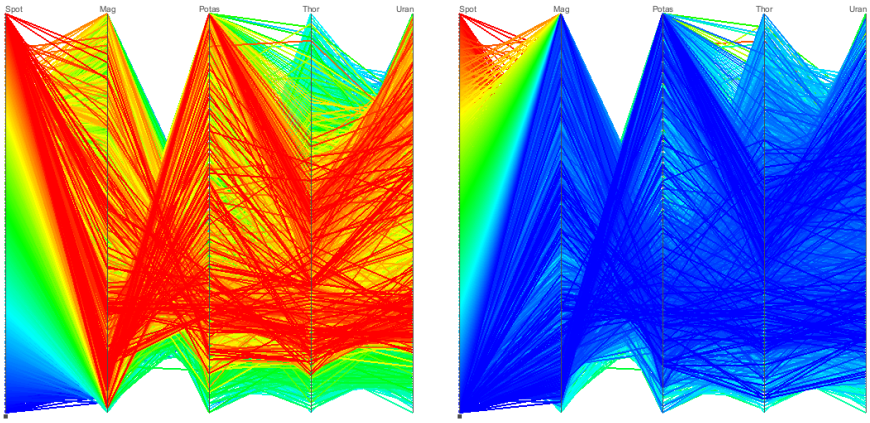
and Nakano [25] developed a 3-D parallel coordinate display that gives each coordinate line a depth proportional to a chosen reference variable. Their approach allows the viewer to visualize a variable's relationships not only to the variables of neighboring axes but also to the reference variable. However, the resulting plots are difficult to parse and require axis reordering and flipping in order for recognizable features to emerge. In this work, we extend the concept of assigning coordinate lines a depth based on a reference variable, and we introduce 3-D lighting cues and automatic viewpoint selection so that relationships can be more easily perceived.

3.2.3 Optimal Views

The task of finding an informative view of a dataset has been well-studied for volume visualization and surface rendering applications. Polonsky et al. [53] give an historical account of this long-standing problem and describe a handful of descriptors used to rank views.

In volume visualization, Bordoloi and Shen [7] assign each voxel in a volume a *noteworthiness* factor and use the concept of information entropy to find the camera location that will maximize the noteworthy content in screen space. Takahashi et al. [58] describe a similar approach that maximizes visibility of surface-level features of differing importance. In both papers, the viewing sphere is uniformly sampled and the optimal viewpoint is selected as the one whose viewport contains the most information according to the importance criteria. Kohlmann et al. [41] concentrate on finding an optimal view for a single user-selected feature. They define optimality according to the feature's shape, orientation, and visibility; the user's viewing history; and the proximity of a view to real-world vantage points.

Our application of optimal view finding does not attempt to choose a single, maximally informative view of a 3-D PCP. Instead, we focus on enabling the user to iteratively navigate through a list of automatically identified features in real-time.



(a) Front (b) Back

Figure 3.1: Two parallel coordinate renderings of the `out5d` dataset, identical in every aspect but the order in which lines are drawn. The lines are colored using a rainbow colormap indexed by the first variable.

3.3 3-D Parallel Coordinates

To reduce occlusion problems in parallel coordinate plots, we expand the traditional 2-D display into a spatially coherent, shaded 3-D display.

3.3.1 Extrusion to 3-D

A primary problem with PCPs is that overplotting obscures coordinate lines, leading to information loss even in small datasets. For example, consider the two renderings in Figure . Nearly all observations with a low value for the first variable are hidden in (a). Conversely, if the lines are plotted in reverse order, as in (b), lines with a high first value are hidden.

If the viewer is interested only in a subset of observations, filtering may be performed to remove unimportant lines. Filtering, by definition, eliminates data from the display, requiring viewers to toggle filters on and off to maintain a global perspective of the data. Such interaction may inhibit the investigative process.

To avoid filtering, we can reduce overplotting by giving each normally 2-D line a depth value and plotting in 3-D. Instead of all lines being drawn in a single plane, the lines span a rectangular prism. To reveal occluded lines, the viewer can rotate the scene without sacrificing data. Figure 3.2 shows this

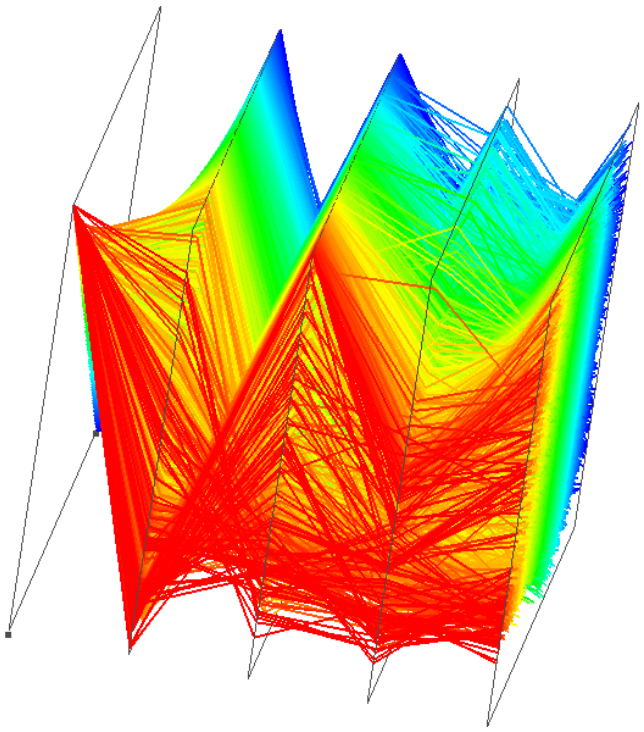


Figure 3.2: An extruded 3-D rendering of the `out5d` dataset. Each line is given a depth value proportional to the magnitude of its first variable. Also, since $v_c = v_d$, color reinforces the depth arrangement.

extrusion applied to the dataset from Figure 3.1.

We assign each observation line's depth value to be proportional to a user-selected variable v_d. Lines are effectively sorted by their v_d values. Lines are colored using a colormap indexed by a user-specified variable v_c. If $v_c = v_d$, color and depth will doubly encode a line's value for that variable, as shown in Figure 3.2. However, the viewer may wish to encode a separate variable using depth.

3.3.2 Shading Coordinate Lines

By extending parallel coordinates into 3-D, we enable users to see occluded lines using rotation instead of filtering. However, 3-D projections like the one shown in Figure 3.2 are difficult to parse. To overcome this problem, we observe that neighboring lines in correlated areas form a surface, which can be shaded with familiar lighting cues.

Traditionally, the underlying geometry of a PCP is a set of polylines and not a surface, making shading difficult. In the work of Wegenkittl et al. [65], surfaces are formed by plotting gaps between coordinate lines as filled polygons. The resulting surfaces can then be shaded using standard illumination techniques. For small, time-variant datasets, this technique is appropriate. But in filling the gaps between coordinate lines, new information is interpolated from the original data, which may mislead the viewer.

To perform surface shading without forming an actual surface, we instead employ an image-space shading technique called *depth shading*, in which a depth map of the scene is treated as a height field to derive each pixel's normal for computing the Phong lighting equation. Figure 3.3(a) shows an example depth map of the `cars` dataset.

In depth shading, a pixel's normal points away from the underlying surface of the depth map, that is, it should be orthogonal to vectors that lie in the surface plane and are directed toward neighboring pixels. We use the neighbors above and to the right of the pixel to calculate these vectors for a pixel at (i, j) with depth $d_{i,j}$, as shown in Eqs. 3.1 and 3.2. The normal at the pixel, then, is just the normalized cross product of these vectors.

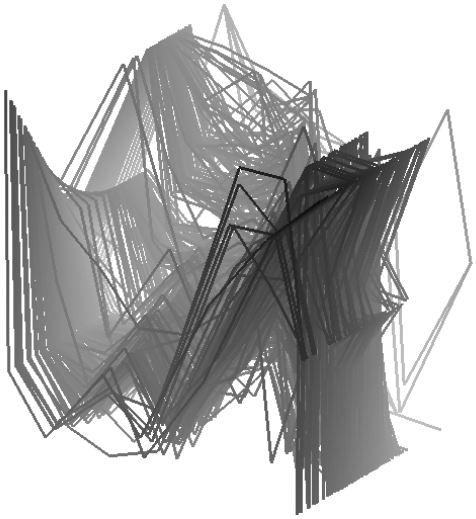
$$v_{\rightarrow} = [1, 0, d_{i+1,j} - d_{i,j}] \tag{3.1}$$

$$v_{\uparrow} = [0, 1, d_{i,j+1} - d_{i,j}] \tag{3.2}$$

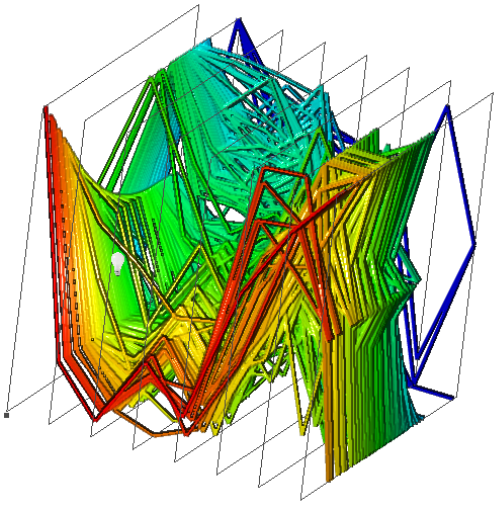
$$\text{normal} = \frac{v_{\rightarrow} \times v_{\uparrow}}{||v_{\rightarrow} \times v_{\uparrow}||} \tag{3.3}$$

To render shaded and extruded parallel coordinates, all lines are rendered to two offscreen buffers, one storing color and one storing depth. An orthographic projection is used exclusively so that relationships between variables are not distorted by perspective. A quadrilateral filling the viewport is then rendered, textured by the offscreen buffers. Each fragment within the quadrilateral is processed by a fragment shader that looks up a fragment's color from the color buffer, calculates its normal using depth shading on the depth buffer, and shades it according to the calculated normal, light position, and color. The result of shading the `cars` dataset is in Figure 3.3(b).

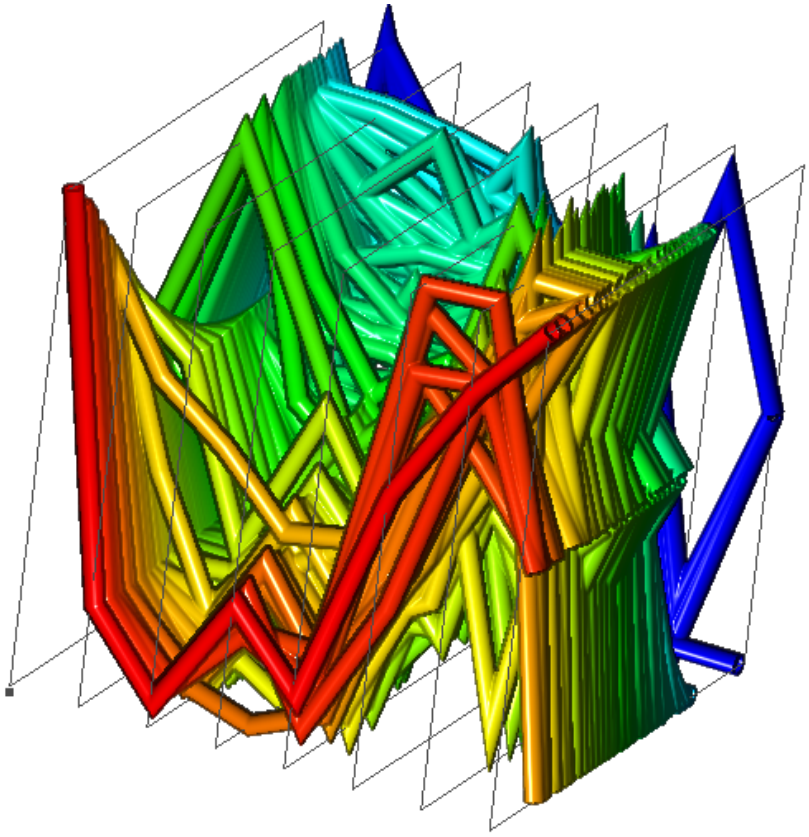
Given that connected regions of parallel coordinate lines may be oriented in a variety of ways, it is essential that the light source be easily positioned



(a) Depth Map



(b) Shaded Lines



(c) Shaded Polycylinders

Figure 3.3: An illustration of depth-shading for 3-D parallel coordinate plots. (a) A rendering of the depth buffer of an extruded 3-D PCP of the `cars` dataset. The magnitude of the gray level indicates distance from the near clipping plane. (b) Application of the depth map from (a) to shade coordinate lines. The 3-D layout of the coordinate lines is made clear through lighting cues.

by the user. We support this positioning by allowing the user to translate a light source icon in the image plane and toward or away from the viewer. In this way, the user can better see through changes in lighting how the surface formed by the coordinate lines bends and in which direction it faces.

Coordinate Line Geometry

Traditionally, coordinate lines have been represented with simple polylines or splines. In moving to 3-D, we have continued to use polylines and have described a method for shading them. However, we can improve the visual quality and lighting cues further by representing each observation with a fully 3-dimensional polycylinder. Figure 3.3(c) shows the `cars` dataset rendered using polycylinders.

With 3-D geometry, one might argue that depth shading is not necessary since normals can be derived from the geometry itself. We choose to retain depth shading, however, because of its convenient haloing effect. Whenever a large jump occurs between neighboring pixels' depths, the calculated normal will point strongly toward the direction of change and less toward the light source. This results in easily distinguished polycylinders and reduced storage and transport costs for the normal data.

3.3.3 Discrete Variables

As described in Section 3.3.1, a coordinate line for observation i is given a third spatial dimension z_i corresponding to a selected variable v_d, as described in Equation 3.4. Here, the dataset is indexed first by an observation identifier and second by a variable identifier.

$$z_i \quad = \quad data[i, v_d] \tag{3.4}$$

If the selected variable has a continuous distribution, the lines will appear adjacent to one another and smoothly sloping surfaces will form in areas of correlation. However, if the selected variable takes on only discrete values, its distribution will contain gaps and lines will be plotted within a limited number of planes. In the `cars` dataset, for example, the variable `origin` has only three possible values, and all lines will be plotted in one of three planes

if this variable is selected.

To overcome the problem of occlusion within discrete variables, we extrude the lines further by allowing another level of extrusion to the depth assignment. The discrete variable v_d is used to partition the unit bounding volume in which the coordinate lines are drawn. Each partition is of equal size P, calculated according to the number of possible values of v_d in Equation 3.6. The user chooses a second variable, v_e, as a means to extrude lines within each partition. We simply extend the assignment of the z-coordinate by translating into the space between v_d's discrete planes according to the magnitude of an observation's variable v_e. The new depth assignment is shown in Equation 3.7, with n being the number of observations in the dataset.

$$V = \{v : v = data'[j, v_d], 0 \leq j < n\} \tag{3.5}$$

$$P = \frac{1}{\text{span}(V) + 1} \tag{3.6}$$

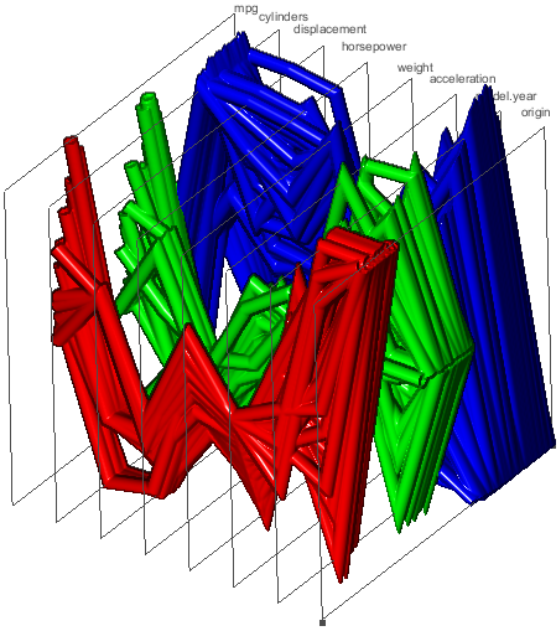
$$z'_i = data[i, v_d] \times (1 - P) + data[i, v_e] \times P \tag{3.7}$$

Assuming the data is normalized and v_d takes on two possible values, a v_d value of 0 and a v_e value of 0.9 for an observation i will place the coordinate line 90% of the distance through the first partition, with $z'_i = 0 \times 0.5 + 0.9 \times 0.5 = 0.45$.

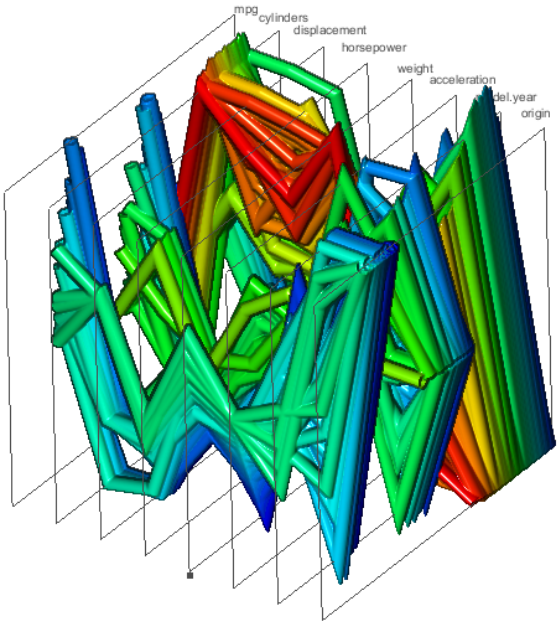
If v_d is not discrete and no partitioning is to be performed, we can still use Equation 3.7 to assign each line a depth value. By simply assigning $v_e = v_d$, lines will be distributed throughout the bounding volume according to the magnitude of v_d. In fact, we can show that for $v_d = v_e$, z'_i is just our original definition of z_i.

$$\begin{aligned} z'_i &= data[i, v_d] \times (1 - P) + data[i, v_d] \times P \\ &= v_d(1 - P + P) \\ &= v_d = z_i \end{aligned}$$

An example of this altered depth assignment is shown in Figure 3.4. Lines are partitioned first according to `origin`. The bounding volume is partitioned into three sections since only three countries of origin exist in the dataset, as shown in (a), where lines are colored according their `origin` values. Within



(a) Origin



(b) Weight

Figure 3.4: Two renderings of the `cars` dataset extruded first by `origin` and second by `weight`. The spatial layout of the renderings is identical, but differing variables are used to index into a rainbow colormap. (a) Colormap indexed by `origin`. (b) Colormap indexed by `weight`.

each partition, lines are sorted according to weight. Since a line's color is determined by a variable independent of the variables used to assign depth, users can choose any variable. By choosing `weight`, it is immediately obvious that one of the three countries produces cars that weigh significantly more than those of the other two.

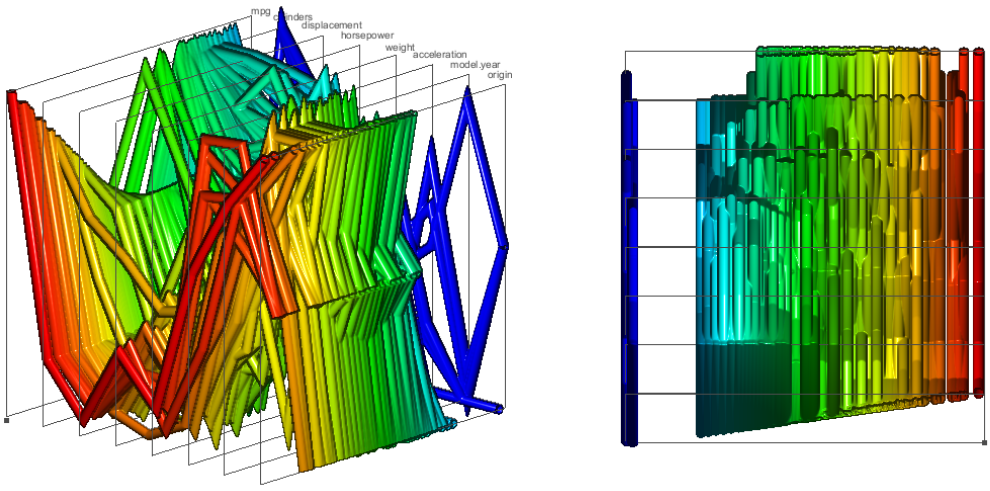
3.4 Optimal Viewpoint Calculation

In extending parallel coordinate displays to 3-D, we have enabled users to interactively rotate the scene to see occluded lines without filtering data. The ultimate goal of data visualization, however, is to find features of interest. With an additional dimension, the task of finding features becomes more complicated for the user. To aid the user in this task, we describe a method to automatically guide users to interesting vantage points.

The first step in automatically guiding users to features is to generate a sufficient number of candidate viewpoints from which to view the plot. We choose to uniformly distribute viewpoints around the viewing sphere. In [41], the authors raise the point that not all vantage points are appropriate for certain features and they allow the viewer to concentrate on views within a subset of the viewing sphere. This may be true for 3-D parallel coordinate plots as well, but we have found that even unconventional sideviews can reveal interesting aspects of the data.

For each view, we generate a color and depth buffer of the plot. The noteworthiness of a view is determined by maximizing a linear combination of image space feature metrics selected and weighted by the user. The color and depth buffers are processed to compute the view's feature vector according to the selected metrics. A feature vector's elements are scaled by user-defined weights and summed, yielding a view's noteworthiness. The collection of viewpoints is then sorted by noteworthiness, and the user can interactively select views from the sorted list.

We now describe a handful of metrics useful for detecting common features of interest.



(a) Largest Footprint

(b) Smallest Footprint

Figure 3.5: Two views of the `cars` dataset selected using footprint size. (a) View containing largest number of filled pixels. (b) View containing smallest number of filled pixels.

3.4.1 Footprint Size

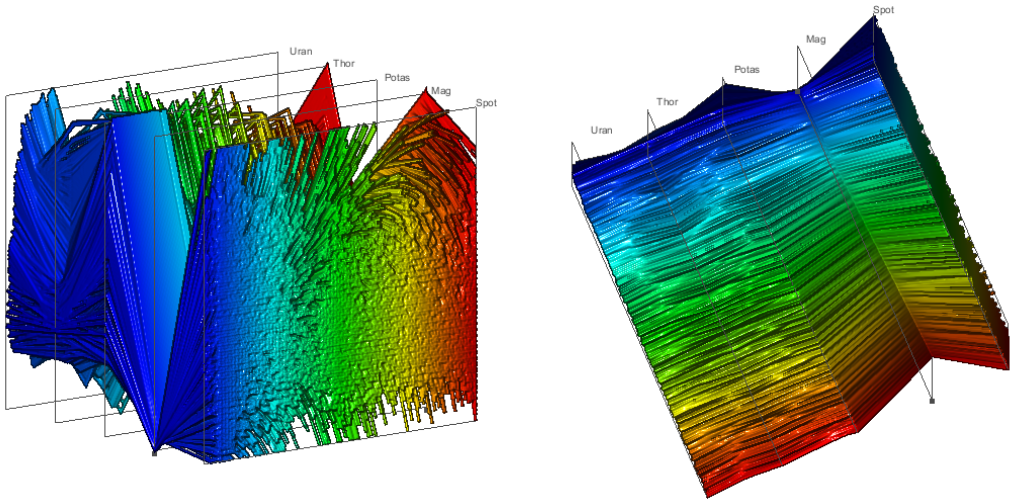
A naive and simple metric for determining whether or not a view is informative is the number of pixels covered by at least one coordinate line. Maximizing this metric will yield a plot with the biggest footprint in the viewport. Figure 3.5 shows a best and worst view of the `cars` dataset based on footprint size.

By itself, this metric is not of striking importance, but it may be combined with other metrics to ensure that a significant portion of the dataset is visible.

3.4.2 Connected Depth Component

If a correlation exists between a subset of observations of two adjacent variables within an interval, coordinate lines will be plotted close to each other, effectively forming a surface. If we choose a view that faces this surface directly, then the depth buffer will contain a region of similar depth values, i.e., a connected component. We can find views that show correlated regions, then, by examining the size of the depth buffer's largest connected component. Large regions of correlation have a large connected component.

Figure 3.6 shows the best and worst view using maximal depth component



(a) Largest Component (b) Smallest Component

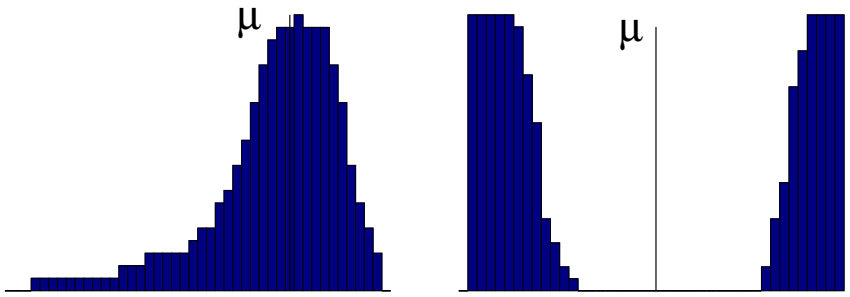
Figure 3.6: Views of the `out5d` dataset ranked according to the size of the largest connected component of the depth buffer. (a) Top-ranking view depicting a region of correlation between magnesium and potassium levels. (b) Worst-ranking view showing no definitive surfaces.

size of the `out5d` dataset. A strong surface forms in (a) in an interval of correlation between magnesium and potassium levels. In (b), no strong surface emerges.

3.4.3 Depth Skewness

Skewness is the measure of a distribution's asymmetry about the mean. In a negatively-skewed distribution, values less than the mean are dispersed more widely than the values greater than the mean, as shown in Figure 3.7(a). If we consider the distribution of values in a view's depth buffer, a negative skewness indicates that the pixels of greatest depth all have similar depth values, while those of the least depth vary widely in their distance from the viewer.

We can interpret negative skewness as an indicator of whether or not a view places outliers in the foreground, since outliers' depths in the scene vary more widely than non-outliers. Figure 3.8 shows two views of a dataset of landmass snow coverage for all latitudes, with some unreported data. Ranking of views is done by combining depth skewness and footprint size, with both metrics of equal weight. Fewer southern latitudes are recorded as having significant snow



(a) Negative Skewness (b) Maximal Variance

Figure 3.7: Two example distributions with mean μ. (a) A distribution with a negative skewness. A view with a negatively skewed depth distribution is likely to keep outliers visible. (b) A distribution with a maximal variance. Views with a maximum grayscale variance likely depict a negative correlation between variables.

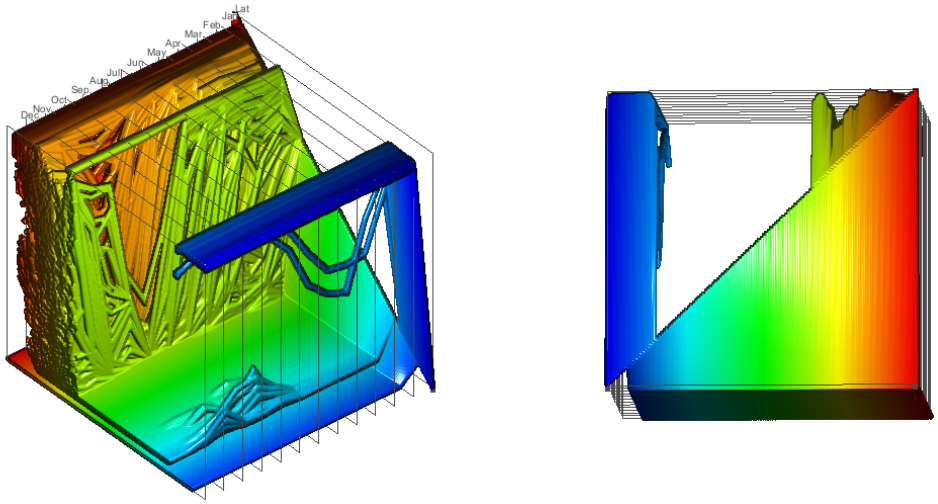
coverage, and these outliers appear prominently in (a).

3.4.4 Gray-value Variance

Surfaces in the 3-D PCP represent regions of correlation only between adjacent variables. Accordingly, an exhaustive search for correlated regions using surface-based metrics like the one described in Section 3.4.3 requires examination of different axis orderings. Since the computational demands of evaluating different orderings is likely to inhibit quick investigation, we describe a different metric that is less sensitive to axis ordering: gray-value variance.

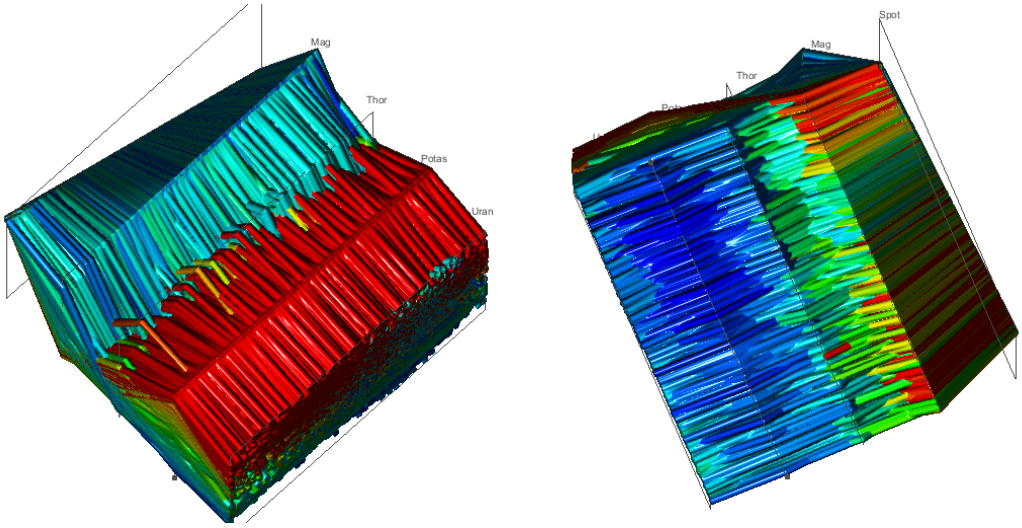
If we use the variable v_c to index into a grayscale colormap, correlations can be investigated by examining the variance of gray-values in the color buffer. If a view has minimal variance, then the dispersion of v_c intensities is least, which might occur if v_c values within a narrow range occur with high frequency. On the other hand, if a view's gray-value distribution has maximal variance, like the distribution shown in Figure 3.7(b), then the dispersion of intensities is greatest and the viewer sees coordinate lines with both low and high v_c values. Axes where lines are colored oppositely from the v_c axis indicate variables negatively correlated to v_c.

An example of an investigation on correlation with potassium content in the `out5d` dataset is shown in Figure 3.9. A view emphasizing the largest spread of intensities appears in (a), in which it's quite apparent that magnesium content



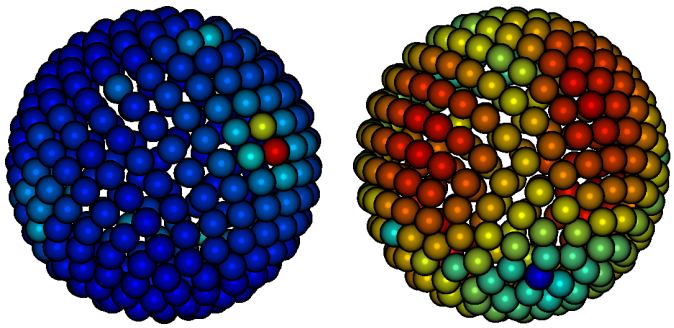
(a) ↓ Skewness + ↑ Footprint (b) ↑ Skewness + ↓ Footprint

Figure 3.8: Two views of a landmass snow coverage dataset. Latitude is used to index into a rainbow colormap, and the remaining 12 axes represent sequential months in one calendar year. Here, the two metrics footprint size and depth skewness are combined to rank views. (a) The best view, showing outliers in the southern latitudes. Using depth skewness alone, this view ranks 14/100. (b) The worst view, having positive skewness and a small footprint.



(a) Negatively Correlated (b) Positively Correlated

Figure 3.9: Two views of the `out5d` dataset selected by gray-value variance. A rainbow colormap maps low potassium levels to blue and high to red. (a) The best view of negative correlation between the non-adjacent magnesium and potassium levels produced by maximizing gray-value variance. (b) The inverse of (a) showing the minimal gray-value variance, i.e., the worst view.



(a) (b)

Figure 3.10: Scores for 300 viewpoints, colored using a rainbow colormap, in which blue maps to a low score and red high. (a) Viewpoints for the `out5d` scored by the size of the largest connected depth component. A high-ranking view is shown in Figure 3.6. (b) Viewpoints of the `cars` dataset, scored by footprint size. Each of the red regions indicates a three-quarter view which maximizes the number of visible pixels.

is negatively correlated to potassium content.

3.4.5 View Sampling

Through informal experimentation, we have found the automatic view selection to return coherent results for neighboring views. Figure 3.10 depicts scores for 300 views of two datasets scored by different metrics. Viewpoints in physical proximity tend to be scored similarly. In examining sample sizes between 50 and 1000 views, we have found 100 viewpoints to give good coverage of the datasets used in this paper without oversaturating the list of detected features with neighboring viewpoints.

3.5 Discussion

Extending PCPs to 3-D does not prohibit traditional interaction found in 2-D plots. For example, though a primary advantage of 3-D PCPs is to reduce occlusion without reducing data, filtering can still be used to hide unimportant data. Axes can be reordered to see how surfaces form between other variable pairs. Visualization of clustered datasets especially benefits from 3-D display: the viewer can sort coordinate lines by a cluster identifier and a second variable

to see how intracluster relationships form. We expect the only changes in migrating traditional 2-D interaction to 3-D to be in adapting the user interface to handle arbitrarily rotated plots.

3.6 Conclusion

We have demonstrated a method of dealing with occlusion in parallel coordinate plots. By moving into three dimensions, we reduce the probability that lines are plotted over one another. Users can interactively rotate the plot to find features without eliminating data. Each coordinate lines is given a depth value according to its variables, making correlations between adjacent variables appear as surfaces. By shading with a conventional lighting model, trends are more easily perceived. We also presented several metrics for automatically finding views of interesting features.

For future work, we plan to investigate using splines instead of polylines as the underlying geometry, which may make the resulting surfaces less noisy in nearly correlated intervals. We also see potential in deriving from the dataset alternative, feature-oriented depth assignments for the coordinate lines.

Chapter 4

Periographs for Large Multivariate Periodic Data

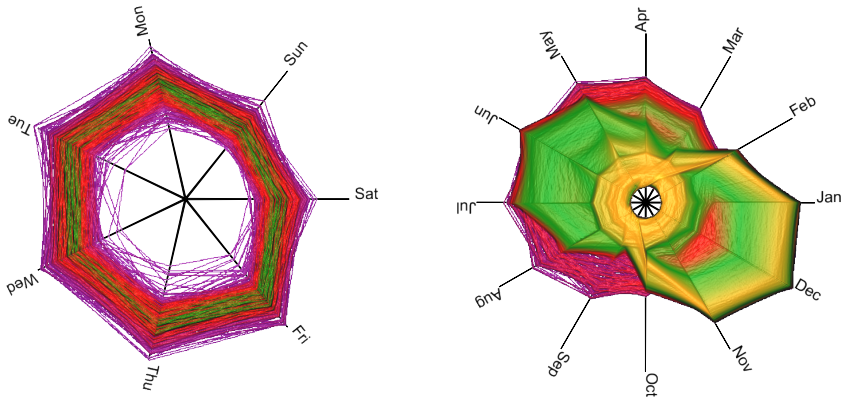
This chapter is a paper currently being reviewed for publication and presentation at a 2009 conference. I am the primary contributor and sole author of this work.

Overview

We present a general-purpose framework for expressing, evaluating, and visualizing boolean range queries on multivariate periodic data. The user is guided in selecting salient intensity ranges through simple interaction with a *periograph*, a frequency plot tailored to reveal trends in cyclical data. Since we compactly represent each location in a data set as an intensity span, range queries are evaluated immediately, and both matching and non-matching locations are visualized according to how well they satisfy a user's query. The utility of our approach is demonstrated using high-dimensional data from a recent climate simulation.

4.1 Introduction

Modern simulations produce data sets with hundreds of variables monitored across time at unprecedented spatial resolutions. Visualization is a natural means of investigating such massive amounts of data, but the data sets' com-



(a) Quebec Births (b) Solar Radiation Reflection

Figure 4.1: Two example periographs depicting cyclic features. (a) A periograph showing the frequencies of the different numbers of infants born per day in Quebec from January 1, 1977, to December 31, 1990. The colormap used is shown in Figure 4.3(b). It is readily apparent that the number of births is approximately normally distributed and that fewer births occur on weekends. (b) A periograph revealing degrees of monthly reflected solar radiation and their frequencies from a 100-year climate simulation.

plexity makes them impractical to visualize in their entirety. Instead, the crucial first step in the visualization process is to determine which portions of the data are relevant to the user. For maximum utility, the user must have some means of expressing and interactively refining such feature queries.

The query framework we describe here enables real-time investigation of multivariate periodic data. Scientists interact with *periographs*, circular histograms customized for revealing trends in periodic data. Large time series data sets are compactly expressed as periographs, from which scientists observe persistence and fluctuation of trends across multiple variables through time and space. Periographs also aid in the formation of boolean range queries, guiding and optionally semi-automatically selecting meaningful intensity ranges for each step in a cycle.

We first outline related work in Section 4.2. Periographs as a standalone visualization method are described in Section 4.3, while in Section 4.4 we apply periographs as a basis for formulating boolean range queries. Section 4.5 exhibits the application of periographs to perform actual queries on climate simulation data, and we close and discuss future work in Section 4.6.

4.2 Related Work

Previous work relevant to the methods described in this paper can be partitioned into two distinct areas: visualization of user queries and visualization of periodic time series data.

4.2.1 Query-driven Visualization

With the amount of data produced by simulations and acquired by capture devices, a data set often cannot be visualized and understood in its raw form. To deemphasize irrelevant data and highlight important data, some means of allowing the user to express feature queries is necessary. A basis for many such queries is the boolean *range query*, in which users express inequalities that observations must meet. For example, to query for locations with low precipitation in winter months, a user might select the query *temperature* $\leq$ *0 and snow coverage* $\leq$ *10%*.

The VisDB project [35] applies visual analysis to the process of querying a database. Each record in the database is assigned a relevance factor according to its degree of matching the user's query, and these factors are visually presented to the user to guide the refinement of query parameters. We employ a similar technique for spatial scientific data.

Range queries are particularly attractive when non-matching data can be eliminated entirely from the visualization process. Stockinger et al. [56] describe an architecture called DEX that accelerates multivariate range query evaluation using an efficient indexing data structure. Gosink et al. [19] introduce a method to study correlation between variables within the results of a boolean range query. In contrast to these two works, we display all locations in a queried data set—in accordance with how nearly they match a query.

Boolean range queries have been applied to study time series data. Glatter et al. [18] use range queries in a distributed computing environment to locate temporal features described using a pattern language inspired by regular expressions.

The Scout framework [47] supports arbitrary query expressions through a data-parallel programming language. Queries are expressed as kernel programs, which are executed directly on graphics hardware for interactive pro-

cessing and display. Current graphics memory limits make this system infeasible without modification for the display of massive time-variant simulation data.

4.2.2 Periodic Time Series Visualization

The circle is a common metaphor for the visualization of periodic time series data. Keim et al. [36] plot multiple time-variant variables on a circular plot, one variable per sector with time progressing per cylinder. Bale et al. [6] similarly decompose a circular plot into sectors, but here a data point is mapped within a sector by treating its day and hour as polar coordinates. Most like our work here are the spiral displays for showing serial periodic data developed by Carlis and Konstan [9] and Weber et al. [64]. A primary difference is that they plot individual data points with time spiraling out at an increasing radius, thereby limiting the amount of data that can be effectively visualized.

A novel method of visualizing and analyzing calendar data is proposed by van Wijk [62], in which data sampled daily is clustered in order to extract cultural and behavioral trends.

A more comprehensive analysis of general time-oriented visualization is offered by Aigner et al. [1]. In addition to providing a survey of visualization for both cyclic and linear time series data sets, the authors posit the importance of data abstraction, clustering, and principal component analysis. Periographs are certainly a form of data abstraction, and they can additionally serve as a basis for visual clustering.

Our approach can be likened to parallel coordinate plots [28, 27] in which each timestep within a cycle is a separate variable. Unlike parallel coordinates, however, we are dealing strictly with massive time series data sets that benefit from an alternate method of representing coordinate lines that is specific to cyclic time and less prone to clutter.

4.3 Periographs

The fundamental primitive of our time-variant query framework is the *periograph*, a frequency plot depicting the periodic tendencies of a single variable in a data set.

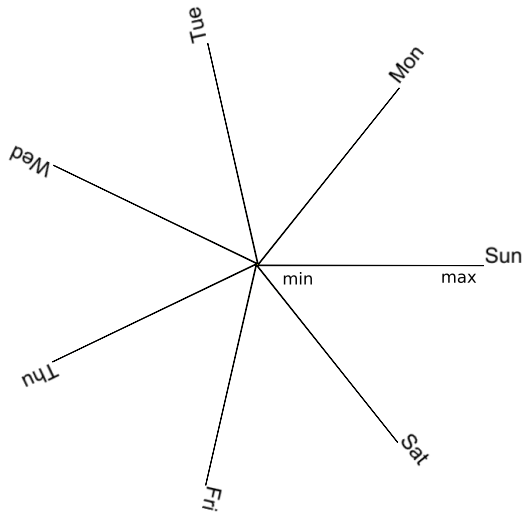


Figure 4.2: The structure of a periograph, with an axis for each timestep in a cycle. Periodic time series data is plotted chronologically across axes. Minimum intensity values are plotted at the center of the periograph, and maximum intensity values at the outer ends of the axes. Henceforth, only axes labels will be drawn to minimize clutter.

4.3.1 Density Plot

Axes are distributed radially around the periograph, one for each timestep in a cycle. For example, the periograph of a data set of birthday frequencies through time may contain an axis for each day of the week. The template for a week-based periograph is illustrated in Figure 4.2, with the plotted data set shown in Figure 4.1(a). All axes meet at the center of the periograph, and this location represents the minimum intensity observed in the variable across all cycles and all periods. The opposite end of an axis represents the maximum observed intensity.

The time series data is plotted to a periograph not unlike coordinate lines are plotted in a parallel coordinates plot. A line is drawn connecting the intensity at the first timestep in a period to the intensity at the subsequent timestep. The last timestep in a cycle is connected to the first timestep of the next cycle, and the process repeats until the last timestep of the last cycle. In this way, the data is "wound" around the periograph.

Formally, we consider a variable as a function of time, $f(t)$. This function is plotted to a periograph as a polyline with a vertex for each discrete timestep t.

A vertex corresponding to t is expressed in polar coordinates as (r_t, θ_t), where

$$\theta_t = \frac{t \bmod \textit{cycle length}}{\textit{cycle length}} \times 2\pi \tag{4.1}$$

$$r_t = \frac{f(t)}{\max\{f(i) : 0 \leq i \leq \textit{maximum timestep}\}}. \tag{4.2}$$

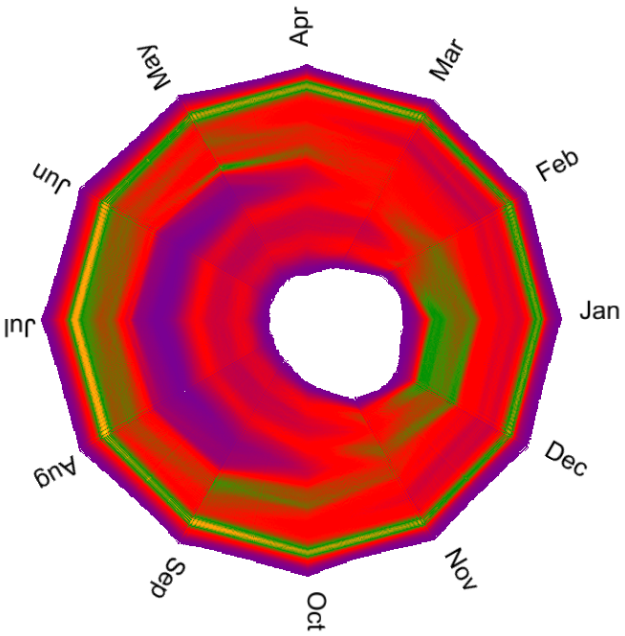
A major limitation of parallel coordinates that we wish to avoid is occlusion of data. As the number of overlapping lines increases in parallel coordinate plots, features become obscured and visual analysis requires some means of data filtering. Our approach avoids the problem of occlusion by foregoing independent treatment of coordinate lines. Instead, we track a pixel's *depth complexity*—the number of times in which the pixel is drawn. The resulting *density plot* [13] gives a fair approximation of dominant temporal trends.

Figure 4.3 shows an example periograph for surface temperature in a 100-year climate simulation. Depth complexity is mapped to color using a colormap of increasing lightness. Since this simulation considers only land and the majority of the globe's landmass is in the northern hemisphere, one can easily see that higher temperatures are more prevalent in May through September, when the northern hemisphere is tilted toward the sun. The low-temperature rings represent the southern polar region, and the low-frequency, high-temperature outer ring signifies an onset of warmer temperatures.

We next apply a standard visualization technique to enhance the periograph. The depth complexities are used to extrude the plot into a heightfield. A Phong lighting model is applied per pixel to emphasize contour changes, and the user can freely orient the plot and translate the light source to emphasize features. Figure 4.4 illustrates this technique applied to the surface temperature periograph.

4.3.2 Multistatistic Periographs

The density plot described above shows only frequency information. From this plot one cannot determine which timesteps or locations contributed to a frequency. To communicate information about such auxiliary variables, we leverage the third dimension that we added in extruding the periograph in



(a) Density Plot

(b) Colormap

Figure 4.3: Unshaded and shaded periographs. (a) The periograph as a density plot. Instead of displaying individual coordinate lines, the number of times a pixel is drawn into is recorded. This depth complexity is used to index into the colormap shown in (b). This colormap is used for every image in this paper.

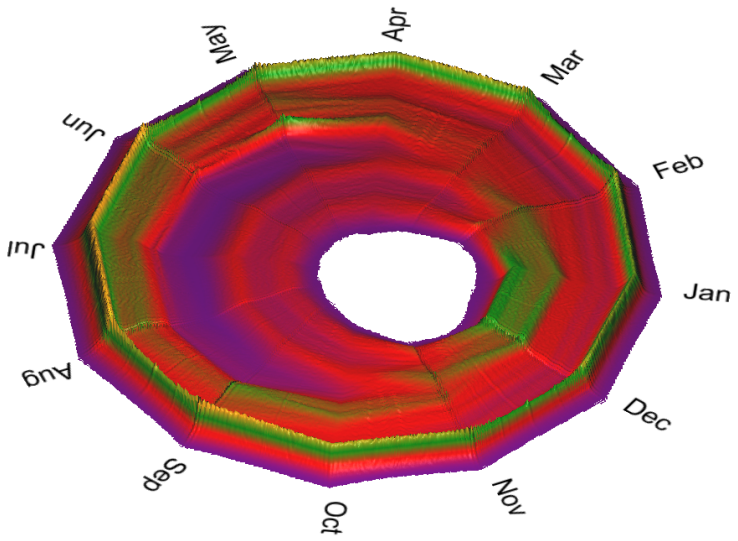
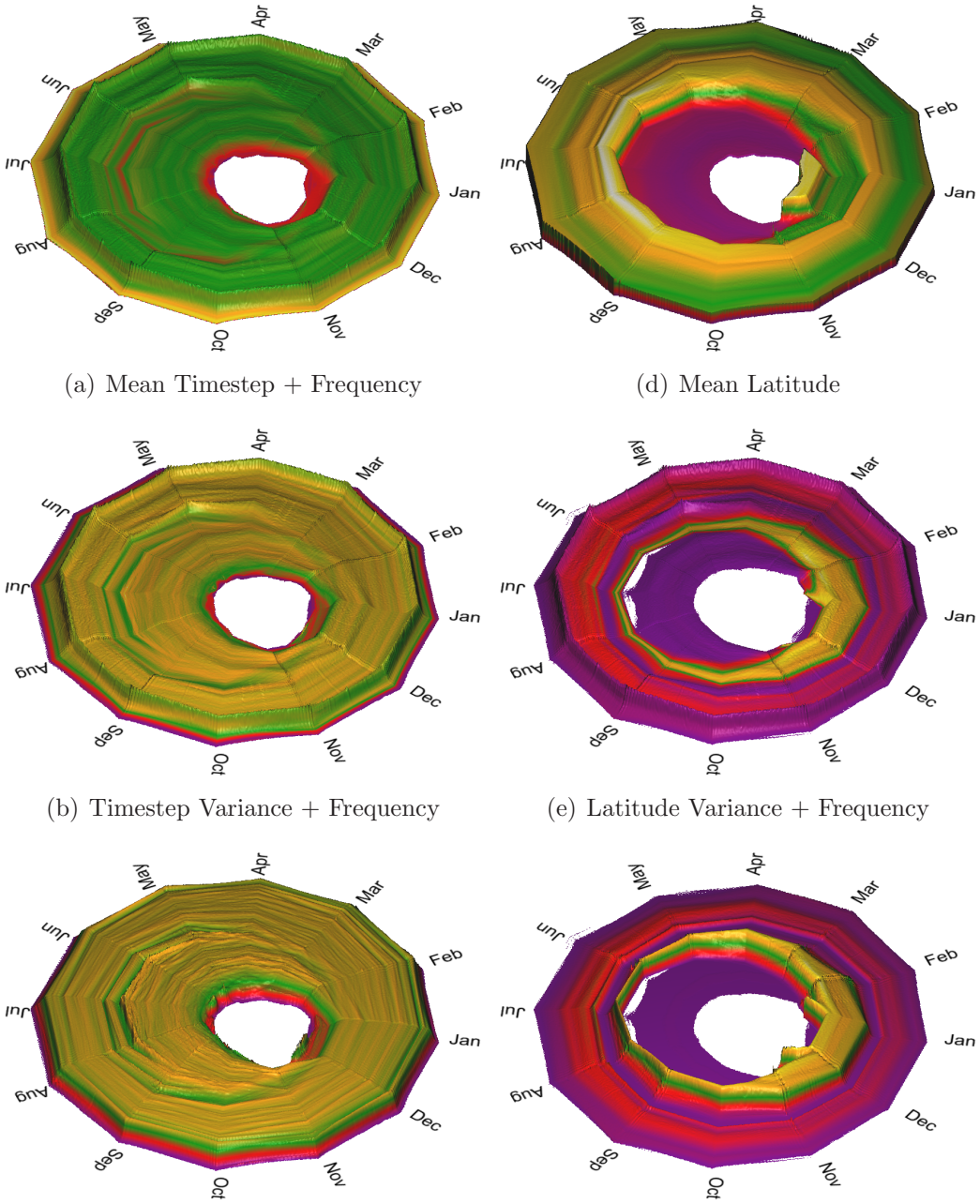


Figure 4.4: By displacing the periograph's height according to depth complexity and applying Phong lighting, trends in frequency are more easily discerned.

order to visualize statistical moments of a pixel's temporal or spatial distribution. For instance, height can be used to depict the average timestep plotted to a pixel and color to depict the frequency at which the pixel's intensity occurs. Or, height can depict the variance in latitudes plotted to a pixel and color the mean timestep.

The current statistics we track for auxiliary variable distributions are mean, variance, and skewness, and users choose at runtime which combination of variables maps to height and color.

We show in Figure 4.5 several examples of how multiple statistics may be visualized. Color is derived from average timestep and height from depth complexity in Figure 4.5(a). For most locations, the timestep distribution is more or less balanced around the middle timestep, which corresponds to green in the colormap. The low-frequency outer ring is colored yellow, indicating a high mean timestep. We can now deduce that these warmer temperatures occur near the end of the 100 years in the simulation. In Figure 4.5(b) color is derived from the variance of a pixel's timestep distribution. We can use this image to corroborate that there is little deviation from the mean at extreme temperatures. Figures 4.5(d) through 4.5(f) use moments from the distribution of latitudes plotted to a pixel.



(a) Mean Timestep + Frequency

(d) Mean Latitude

(b) Timestep Variance + Frequency

(e) Latitude Variance + Frequency

(c) Timestep Variance

(f) Latitude Variance

Figure 4.5: Several multistatistic periographs, in which color depicts one metric and height optionally depicts another. Metrics currently supported include an intensity's frequency and mean, variance, and skewness of auxiliary variables' distributions. (a) Mean timestep → color, frequency → height. (b) Timestep variance → color, frequency → height. (c) Timestep variance → color, timestep variance → height. (d) Mean latitude → color, mean latitude → height. (e) Latitude variance → color, frequency → height. (f) Latitude variance → color, latitude variance → height.

4.4 Periodic Queries

Periographs in and of themselves are a useful means of studying trends in periodic time series data, but in plotting a data set to a periograph, spatial information is lost. However, the periograph can be used in combination with traditional space-preserving rendering methods. In particular, we can use the periograph as a canvas for multivariate querying and present the evaluated query in a slice viewer or volume renderer. To simplify our following explanation, we first consider single variable range queries and expand our method to multivariate queries in Section 4.4.3.

4.4.1 Multivariate Range Query Design

Histograms have long been used in the design of transfer functions, and we similarly consider a periograph a logical basis for designing queries on periodic data. Multivariate range queries have been demonstrated to be an effective means of extracting and visualizing relevant subsets of data, and periographs are a natural means of assisting a user as he chooses the parameters of a range query.

To enable users to form range queries on periodic data, we augment the periograph to include range selectors on each axis of the periograph. Using a mouse, users can click and drag to modify the range of intensities that are selected for each timestep in a cycle. With the periograph as a backdrop, an informed decision can be made about the frequency of intensities being selected and the moments of auxiliary variables' distributions. Figure 4.6 shows an example of the augmented periograph in which the user is querying to see which locations contribute to the ring of low surface temperatures visible in the periograph.

Design Assistance

Depending on the number of timesteps in a cycle, modifying range selections can be tedious for certain operations. We have found it useful to incorporate several mechanisms for faster range selection.

The first mechanism is a global range selection. The user can specify that the range selection being applied to the current axis should apply to all axes.

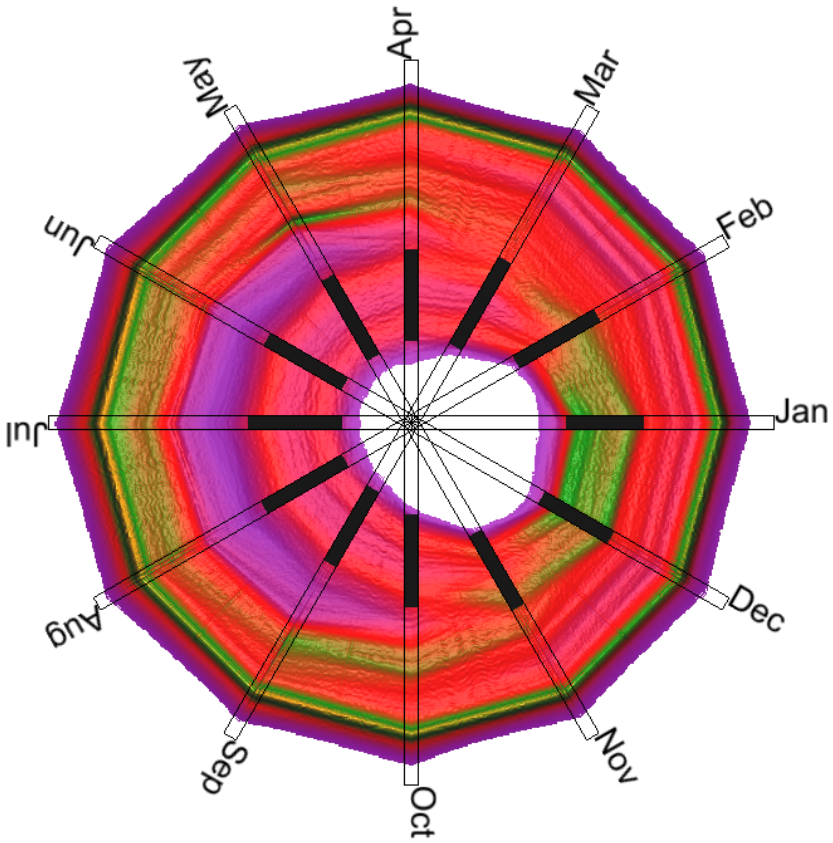


Figure 4.6: The periodic range query designer. Users select intensity ranges by manipulating gauges for each step in the cycle with the aid of the periograph. If a particular step doesn't pertain to a query, then any intensity at the step is valid and the entire range is selected. In this example, the user has selected ranges that isolate a year-round high-frequency feature of low surface temperatures.

In this way, queries for constant intensity can be formed without having to repeat the same mouse interaction for every axis in the cycle. Such a query might be useful in order to find locations where vegetation is present year round or wind speed is always minimal.

The second mechanism is motivated out of the common practice of selecting a range of values comprising a mode in the distribution. If the user clicks on an axis without dragging the mouse, the extent of the range is automatically calculated based on the derivative of the axis' univariate distribution. The range is extended both outward and inward from the selected intensity until local minima are reached in the distribution.

4.4.2 Query Evaluation

If there are C timesteps in a cycle, the resulting query is a set of exactly C ranges. We evaluate a query by determining how closely each location is to falling within the selected intensity ranges. If a location satisfies every range selection at every step in a cycle, it is assigned a *distance* of 0. Otherwise, its distance is a summation of the minimal differences between the location's intensities and the query ranges. In other words, for a particular step in the cycle, distance is calculated using a location's intensity closest to falling within the range for that step.

Figure 4.7 illustrates all possible classes of arrangements between a location's normalized intensity range and a query range. In only two cases is there no overlap, where distance is non-zero. A location satisfies a query range exactly, then, if its intensity range overlaps the query range in some manner.

We use this method of calculating distances because it allows the query to be evaluated at interactive rates. Instead of examining every timestep for every location, we can store as a preprocessing step only the minimum and maximum values that a location takes on for each step in the cycle. For the climate simulation data set, this means we store 24 values for each variable at each location, 12 minima and 12 maxima. This reduces the data size by $\frac{1200-24}{1200} = 98\%$. The minimum value for location x at step i in the cycle is $f_{min}(x, i)$, and the maximum is $f_{max}(x, i)$.

Formally, the distance for location x is defined according to Equation 4.3. Note that the range bounds are in $[0, 1]$, representing the percent offset be-

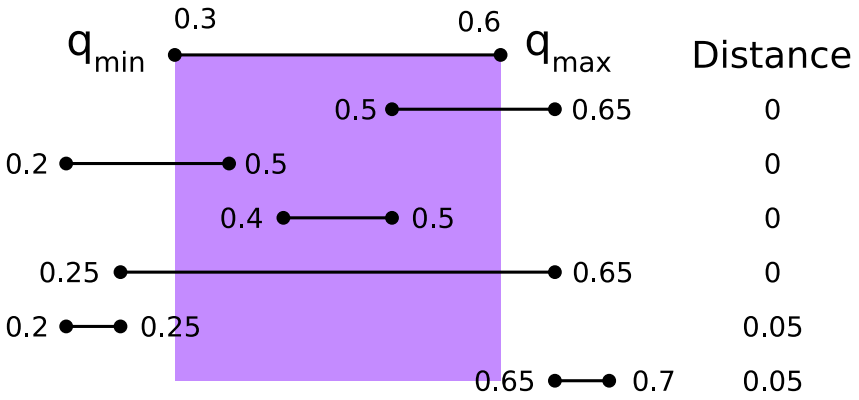


Figure 4.7: Example arrangements between a query range and a location's span of intensities. Six arrangements are possible, two of which involve no overlap. In these two cases a location is not considered to match the query. A non-matching location is assigned a non-zero distance according to the gap between the query range and its intensity span.

tween the minimum and maximum values observed in the data set. Only one of the two terms within the summation will ever be non-zero, and only then if the intensity range relates to the query range in one of the two non-overlapping arrangements shown in Figure 4.7. To calculate a location's score in Equation 4.5, we normalize the distance for a location by dividing it by the maximum possible distance, which is a function only of the set of query ranges. It's computed in Equation 4.4 as the sum of maximum conceivable distances an intensity range may have from the query range, though this distance may not actually occur for a given data set and query.

$$distance_x = \sum_{i}^{C} \max(0, q_{min}(i) - f_{max}(x,i)) + \max(0, f_{min}(x,i) - q_{max}(i)) \tag{4.3}$$

$$max\ distance = \sum_{i}^{C} \max(q_{min}(i), 1 - q_{max}(i)) \tag{4.4}$$

$$score_x = \frac{distance_x}{max\ distance} \tag{4.5}$$

After scores have been computed for each location, the result can be visualized in a standard slice viewer or volume renderer. In this work, we focus on 2-D climate data. Periodic volume data is not currently in common usage,

but periograph queries extend to 3-D without modification. To visualize how a location meets a periodic range query, its score is used to index into a colormap. The optical properties of the evaluated query can be altered through traditional transfer function design widgets.

4.4.3 Multivariate Queries

Supporting multidimensional queries is a straightforward extension of the methods described above. For the set of variables V under consideration, $|V|$ periographs are displayed and a query produces $C \times |V|$ ranges. A multivariate query is evaluated by a simple extension to Equations 4.3 and 4.4, where $q_{\{min,max\}}$ and $f_{\{min,max\}}$ have been expanded to include the variable under examination. These changes are reflected in Equations 4.6 and 4.7.

$$distance_x = \sum_{v \in V} \sum_{i}^{C} \max(0, q_{min,v}(i) - f_{max,v}(x,i)) + \max(0, f_{min,v}(x,i) - q_{max,v}(i)) \tag{4.6}$$

$$max\ distance = \sum_{v \in V} \sum_{i}^{C} \max(q_{min,v}(i), 1 - q_{max,v}(i)) \tag{4.7}$$

4.5 Discussion

Our interactive query framework was developed in order to investigate climate simulation data, though it can be applied to any periodic data. We examine how scientists might use our tools to explore a 1200-month (January 2000–December 2099) simulation coordinated by the Intergovernmental Panel on Climate Change. The data set is comprised of 61 variables measured at points of landmass across a 256×128 cylindrical projection of the Earth's surface. Values represent monthly averages.

To gain a comprehensive overview of all 61 variables, the user can examine the periographs *en masse* as shown in Figure 4.11. Through this overview, interrelated variables can be visually clustered according to their cyclic trends.

Using the overview as a starting point, the user can then view individual periographs to investigate variables in isolation. For example, Figure 4.8 shows a periograph and query on the monthly average amount of reflected solar ra-

diation. Two noticeable "flares" of high frequencies appear in the periograph. From the periograph, it is apparent that these flares coincide with summer and winter solstices. A query is formed around the December-based flare, and the evaluation of the query in Figure (b) reveals the snow-covered and desert locations that contribute to the high degree of reflection.

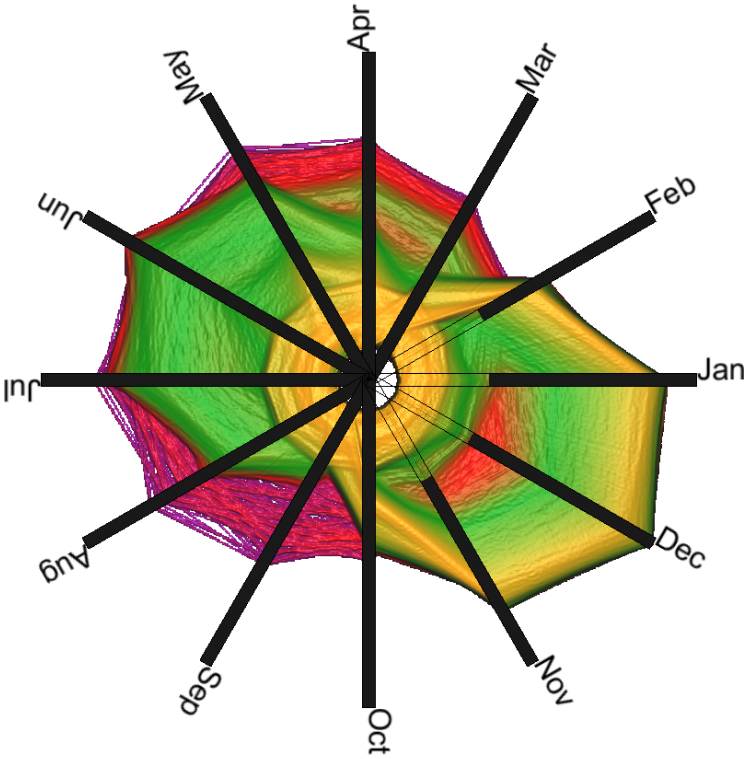
Snow coverage is examined in Figure 4.9. In the frequency periograph, distinct ridges can be seen where many locations' percentage of landmass covered in snow rises or falls drastically between months. The query in Figure (b), however, is for locations where March snow coverage is high at some timestep during the simulation, but February snow coverage is low at some timestep. The query reveals strips in the northern hemisphere where warming temperatures are causing milder winters.

A multivariate query is expressed in Figure 4.10. The two variables considered here are the amount of surface runoff and rainfall. The periograph of surface runoff in (a) reveals heightened levels of runoff in May through August. The periograph for rainfall in (b) exhibits a similar trend, including some outliers due to heavy February rains in Brasil, Madagascar, and Indonesia. The query pinpoints locations where an average amount of rainfall occurs with above average surface runoff. The prime matching locations are in northern Canada and Alaska, where summer glacial melt cannot be fully infiltrated by the soil.

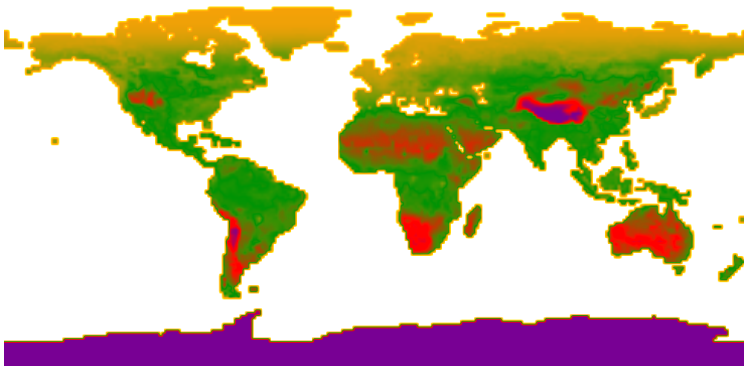
4.6 Conclusion

We have presented a practical and interactive framework for generating and evaluating range queries on periodic data. Through the use of a frequency plot customized for revealing cyclical trends, users are informed about the statistical properties of the underlying time series in a concise manner. Using the periograph as a guide, meaningful queries can be expressed quickly and interactively modified using the feedback from the instantaneous evaluation. The utility of our approach is demonstrated in the context of a recent climate simulation.

For future work, we intend to expand the information that is communicated by a periograph-based query. In particular, we would like to reveal specific time

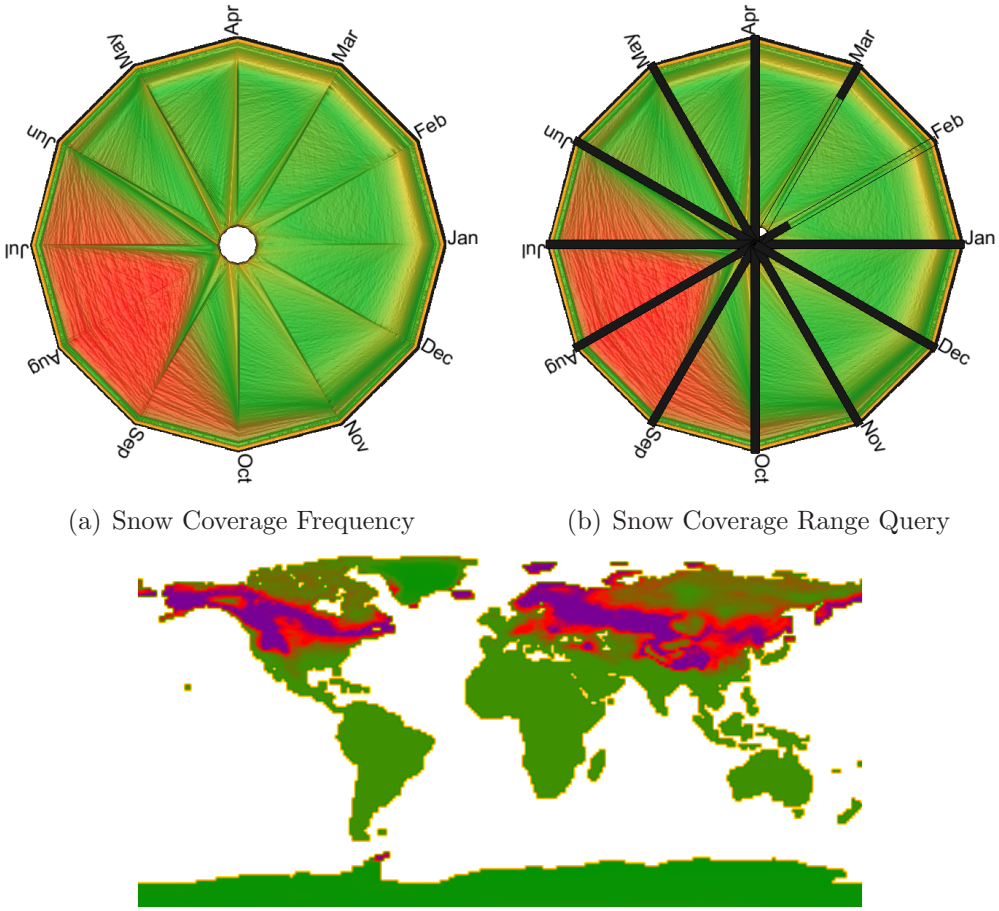


(a) Reflected Solar Radiation Frequency



(b) Evaluated Query

Figure 4.8: A query on high-intensity, high-frequency reflected solar radiation. (a) A periograph showing marked high-frequency, seasonal features of reflected solar radiation. (b) On evaluating the query, Antarctica and snow-covered, high-elevation locations are revealed as primary contributors to the high-frequency feature in the periograph.

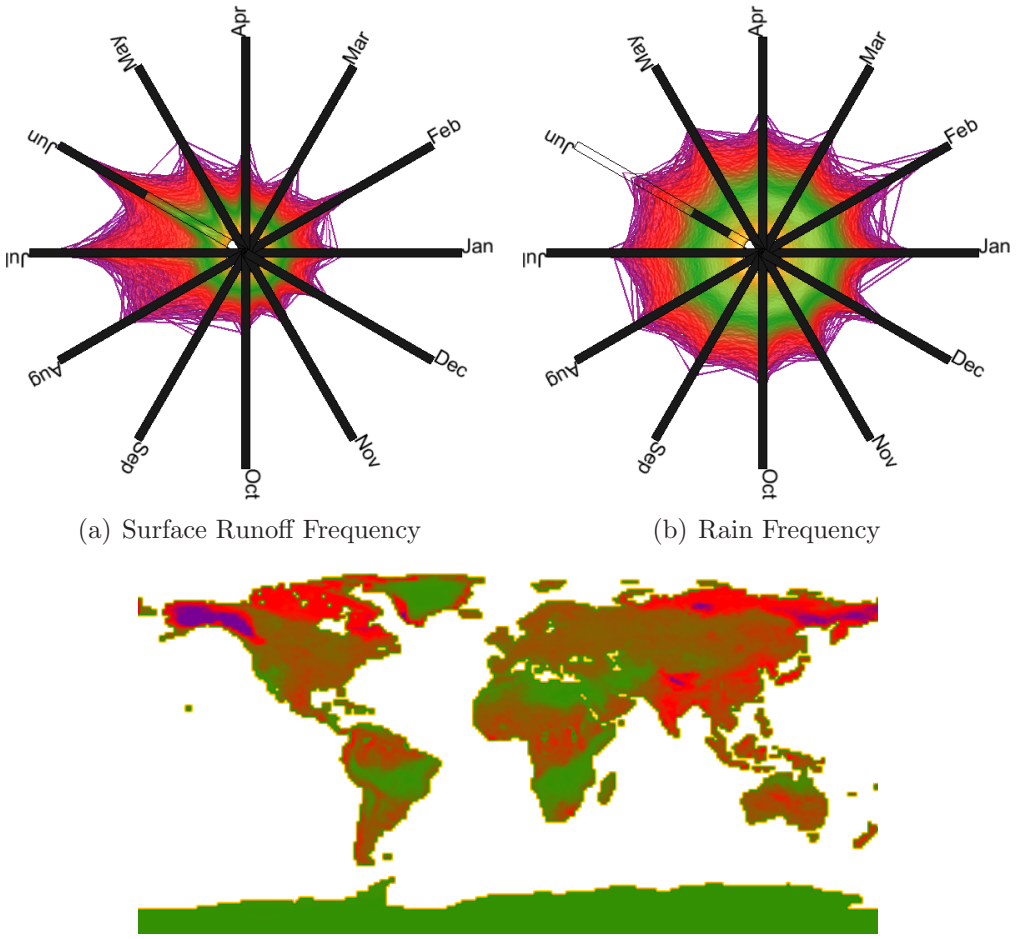


(a) Snow Coverage Frequency

(b) Snow Coverage Range Query

(c) Evaluated Query

Figure 4.9: Periographs and their application to range query design. (a) A periograph showing frequencies of percentages of landmass covered in snow. (b) A range query searching for locations that experience less snow in some February than they do in some March. (c) Locations that are projected to experience warming trends that significantly reduce winter snow coverage.



(a) Surface Runoff Frequency

(b) Rain Frequency

(c) Evaluated Query

Figure 4.10: A multivariate range query. This query considers (a) surface runoff and (b) rainfall. Here, the user is searching for locations where the intensity of runoff in June is high but the amount of rainfall is typical. (c) Matching locations occur in regions where summer glacial melt cannot be fully infiltrated by the soil.

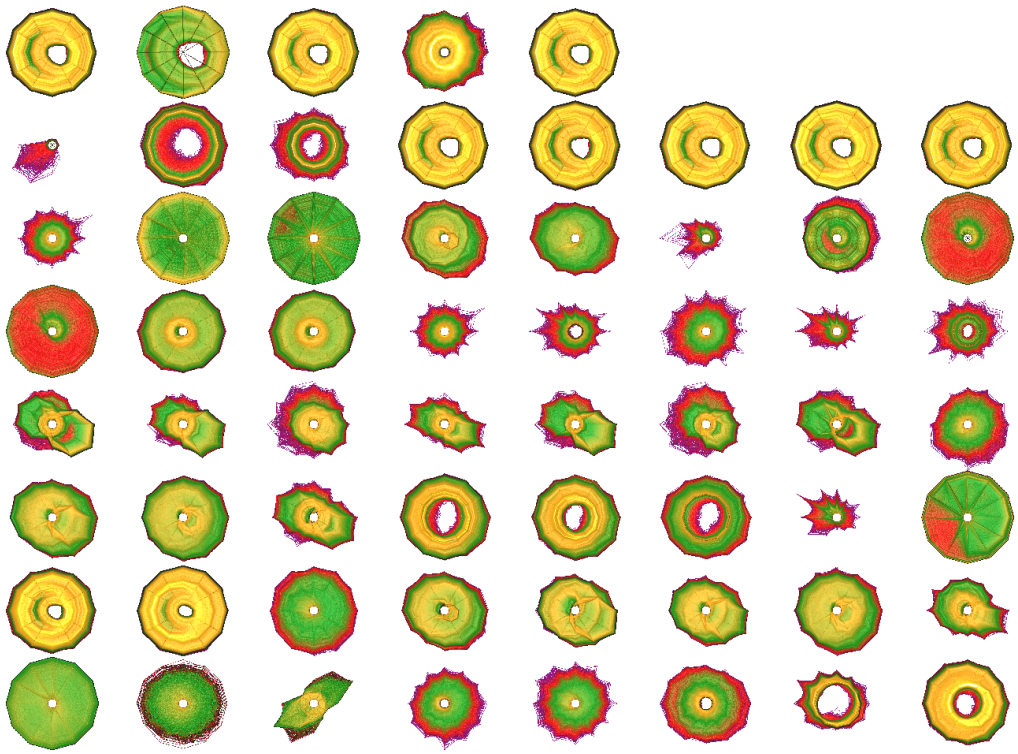


Figure 4.11: A snapshot of periographs for all 61 variables of the IPCC climate simulate data set. This overview is exploited by the user to quickly cluster interdependent variables and detect overall trends. These periographs compactly represent all 1200 timesteps of all variables. To avoid saturation due to several extremely high-frequency features, frequencies have been log-transformed before being used to index into the colormap.

information regarding when a location matches the selected intensity ranges. Periographs also are a medium in which clustering and segmentation may be performed quickly in image space, which may benefit analysis of complex data sets.

Acknowledgements

The births data set from Figure 4.1(a) is courtesy R.J. Hyndman.

Conclusion

> On Exactitude in Science . . . In that Empire, the Art of Cartography attained such Perfection that the map of a single Province occupied the entirety of a City, and the map of the Empire, the entirety of a Province. In time, those Unconscionable Maps no longer satisfied, and the Cartographers Guilds struck a Map of the Empire whose size was that of the Empire, and which coincided point for point with it. The following Generations, who were not so fond of the Study of Cartography as their Forebears had been, saw that that vast Map was Useless, and not without some Pitilessness was it, that they delivered it up to the Inclemencies of Sun and Winters. In the Deserts of the West, still today, there are Tattered Ruins of that Map, inhabited by Animals and Beggars; in all the Land there is no other Relic of the Disciplines of Geography.
>
> Suarez Miranda, Viajes de varones prudentes, Libro IV, Cap. XLV, Lerida, 1658

In this work we have described several methods which exploit frequency information to better identify features in visualized data. For data in which a spatio-temporal neighborhood can be composed, neighborhood distribution queries allow the user to describe features in terms of probabilistic statements. To gain insight into high-dimensional aspatial data, 3-D parallel plots were introduced, with a means of automatic detection of compelling viewpoints based on image-space distribution metrics. Lastly, for temporal data, a new

cyclic frequency histogram called a periograph was developed as a means for easy detection of periodic features and the design of boolean range queries.

These three approaches have the common goal of bringing the vocabulary of user queries closer to everyday human reasoning. Much of human decision-making rests on our understanding of how social and physical systems work. We acquire many measurements that describe these systems, but we cannot make sense of raw data without abstracting it to manageable summary information. Campaigning politicians want to know which demographic will maximize their chances of winning an election, manufacturing industries want to know the source of tainted or defective products in order to minimize losses, climatologists want to determine the factors that govern climate change to prevent ecological devastation, and consumers want to balance complex systems of costs and benefits before they make big purchases. Such decisions are made by examining high-frequency trends, outliers, and correlations and causations between variables. We look for patterns in our data to formulate hypotheses, which we correct as we investigate instances where the pattern is violated.

To be effective, our visualization tools must support this probabilistic vocabulary which we use everyday. For very simple data sets, a simple graphical display of data allows the human eye and brain to automatically detect patterns and perform abstraction. For more complicated multivariate and temporal data, however, making frequency-based decisions requires more sophisticated tools. As such, the methods described in this work exploit frequency information in independent and complementary ways. Periographs are used to quickly detect high-frequency cyclic trends in temporal data. They are constructed without assistance from the user and can be interpreted as any histogram can. In contrast, viewpoint distributions for 3-D parallel coordinate plots do not provide summary information in and of themselves and are not directly visualized, but they are used to semiautomatically identify viewpoints that depict interesting features. The criteria that makes a feature interesting is formulated using statistical moments from the image space distribution of colors and depth values. In further contrast, neighborhood queries are entirely dependent on the feature definitions expressed by the user. SeeDQ allows the user to search for region-based features of arbitrary definition. The query tool is closely integrated with the resulting visualization so that the user receives

proper feedback to refine the query and interpret results.

In closing, we have demonstrated the utility of incorporating higher-level frequency information to the investigation of real world datasets. All work has been or is in the process of being shared with the broader scientific community. Collaborations with domain scientists are growing strong as data is much more cheaply acquired or simulated, and the accompanying need to make sense of that data will require tools such as these to allow users to query for whatever features they find interesting.

Future work in the area of distribution-based query visualization will be essential as computational science further embraces visualization as both a means of communicating findings and for performing interactive analysis. The limitations of processor and bus speeds and memory capacity necessitate an accelerated means of dealing with data, and distributions offer a more concise model that can be manipulated at interactive rates. They additionally have widespread application and can easily be adopted to meet the diverse needs of scientists.

Bibliography

Bibliography

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

[2] Hiroshi Akiba, Nathaniel Fout, and Kwan-Liu Ma. Simultaneous classification of time-varying volume data based on the time histogram. In *EuroVis*, pages 171–178, 2006.

[3] Hiroshi Akiba, Kwan-Liu Ma, Jacqueline H. Chen, and Evatt R. Hawkes. Visualizing multivariate volume data from turbulent combustion simulations. *Computing in Science and Engineering*, 9(2):76–83, 2007.

[4] Almir Olivette Artero, Maria Cristina Ferreira de Oliveira, and Haim Levkowitz. Uncovering clusters in crowded parallel coordinates visualizations. In *Proceedings of IEEE Symposium on Information Visualization*, pages 81–88, Washington, DC, USA, 2004. IEEE Computer Society.

[5] Chandrajit L. Bajaj, Valerio Pascucci, and Daniel R. Schikore. The contour spectrum. In *Proceedings of IEEE Visualization*, pages 167–ff., Los Alamitos, CA, USA, 1997. IEEE Computer Society Press.

[6] Kim Bale, Paul Chapman, Nick Barraclough, Jon Purdy, Nizamettin Aydin, and Paul Dark. Kaleidomaps: a new technique for the visualization of multivariate time-series data. *Information Visualization*, 6(2):155–167, 2007.

[7] Udeepta D. Bordoloi and Han-Wei Shen. View selection for volume rendering. *Proceedings of IEEE Visualization*, 00:62, 2005.

[8] Wenli Cai and Georgios Sakas. Data intermixing and multi-volume rendering. *Computer Graphics Forum*, 18(3):359–368, 1999.

[9] John V. Carlis and Joseph A. Konstan. Interactive visualization of serial periodic data. In *UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology*, pages 29–38, New York, NY, USA, 1998. ACM.

[10] Roger Crawfis and Nelson Max. Texture splats for 3d vector and scalar field visualization. In *Proceedings of IEEE Visualization*, pages 261–266, October 1993.

[11] Helmut Doleisch, Martin Gasser, and Helwig Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *VISSYM '03: Proceedings of the Symposium on Data Visualisation 2003*, pages 239–248, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[12] Robert A. Drebin, Loren Carpenter, and Pat Hanrahan. Volume rendering. In *Computer Graphics (Proceedings of SIGGRAPH 88)*, volume 22, pages 65–74, August 1988.

[13] John Miller Edward, John J. Miller, and Edward J. Wegman. Construction of line densities for parallel coordinate plots. In *Computing and Graphics in Statistics*, pages 107–123. Springer, 1991.

[14] Geoffrey Ellis and Alan Dix. Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):717–724, 2006.

[15] Elena Fanea, Sheelagh Carpendale, and Tobias Isenberg. An interactive 3d integration of parallel coordinates and star glyphs. In *Proceedings of IEEE Symposium on Information Visualization*, page 20, Washington, DC, USA, 2005. IEEE Computer Society.

[16] Shiaofen Fang, Tom Biddlecome, and Mihran Tuceryan. Image-based transfer function design for data exploration in volume visualization. In *Proceedings of IEEE Visualization*, pages 319–326, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[17] Ying-Huey Fua, Matthew O. Ward, and Elke A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of IEEE Visualization*, Washington, DC, USA, 1999. IEEE Computer Society.

[18] Markus Glatter, Jian Huang, Sean Ahern, Jamison Daniel, and Aidong Lu. Visualizing temporal patterns in large multivariate data using textual pattern matching. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1467–1474, 2008.

[19] Luke Gosink, John Anderson, Wes Bethel, and Kenneth Joy. Variable interactions in query-driven visualization. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1400–1407, 2007.

[20] Martin Graham and Jessie Kennedy. Using curves to enhance parallel coordinate visualisations. In *Proceedings of IEEE Symposium on Information Visualization*, page 10, Washington, DC, USA, 2003. IEEE Computer Society.

[21] Markus Hadwiger, Christoph Berger, and Helwig Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proceedings of IEEE Visualization*, page 40, Washington, DC, USA, 2003. IEEE Computer Society.

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

[23] Jiří Hladůvka, Andreas König, and Eduard Gröller. Salient representation of volume data. In D. Ebert, J. M. Favre, and R. Peikert, editors, *Data Visualization 2001, Proceedings of the Joint Eurographics – IEEE TVCG Symposium on Visualization*, pages 203–211,351, 2001.

[24] Forrest M. Hoffman, William W. Hargrove, David J. Erickson, and Robert J. Oglesby. Using clustered climate regimes to analyze and compare predictions from fully coupled general circulation models. *Earth Interactions*, 9(10):1–27, August 2005. doi:10.1175/EI110.1.

[25] Keisuke Honda and Junji Nakano. 3 dimensional parallel coordinates plots and its use for variable selection. In *Computational Statistics (COMPSTAT 2006)*, pages 187–195, 2006.

[26] Peter Imrich, Klaus Mueller, Dan Imre, Alla Zelenyuk, and Wei Zhu. 3D ThemeRiver. *Poster at: IEEE Information Visualization Symposium*, 2003.

[27] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proceedings of IEEE Visualization*, pages 361–378, 1990.

[28] Alfred Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(2):69–91, 1985.

[29] Heike Jänicke, Alexander Wiebel, Gerik Scheuermann, and Wolfgang Kollmann. Multifield visualization using local statistical complexity. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1384–1391, 2007.

[30] Jimmy Johansson, Matthew Cooper, and Mikael Jern. 3-dimensional display for clustered multi-relational parallel coordinates. In *IV '05: Proceedings of the Ninth International Conference on Information Visualisation*, pages 188–193, Washington, DC, USA, 2005. IEEE Computer Society.

[31] Jimmy Johansson, Patric Ljung, Mikael Jern, and Matthew Cooper. Revealing structure within clustered parallel coordinates displays. In *Proceedings of IEEE Symposium on Information Visualization*, page 17, Washington, DC, USA, 2005. IEEE Computer Society.

[32] C. Ryan Johnson, Markus Glatter, Wesley Kendall, Jian Huang, and Forrest Hoffman. Querying for feature extraction and visualization in climate modeling. In *ICCS 2009: Geo Computation*, volume 5545 of *Lecture Notes in Computer Science*, pages 416–425. Springer, 2009.

[33] C. Ryan Johnson and Jian Huang. Distribution driven visualization of volume data. *IEEE Transactions on Visualization and Computer Graphics*, 15(5):734–746, 2009.

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

[35] D. A. Keim and H. P. Kriegel. VisDB: database exploration using multidimensional visualization. *Computer Graphics and Applications, IEEE*, 14(5):40–49, 1994.

[36] Daniel A. Keim, Jörn Schneidewind, and Mike Sips. Circleview: a new approach for visualizing time-related multidimensional data sets. In *AVI '04: Proceedings of the working conference on Advanced visual interfaces*, pages 179–182, New York, NY, USA, 2004. ACM.

[37] Gordon Kindlmann and James W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *IEEE Symposium on Volume Visualization*, pages 79–86, 1998.

[38] Gordon Kindlmann, Ross Whitaker, Tolga Tasdizen, and Torsten Möller. Curvature-based transfer functions for direct volume rendering: Methods and applications. In *Proceedings of IEEE Visualization*, pages 513–520, 2003.

[39] Joe Kniss, Gordon Kindlmann, and Charles Hansen. Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 8(3):270–285, July-September 2002.

[40] Joe M. Kniss, Robert Van Uitert, Abraham Stephens, Guo-Shi Li, Tolga Tasdizen, and Charles Hansen. Statistically quantitative volume visualization. In *Proceedings of IEEE Visualization*, volume 00, page 37, Los Alamitos, CA, USA, 2005. IEEE Computer Society.

[41] Peter Kohlmann, Stefan Bruckner, Armin Kanitsar, , and Eduard Gröller. Livesync: Deformed viewing spheres for knowledge-based navigation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1544–1551, 2007.

[42] David H. Laidlaw, Kurt W. Fleischer, and Alan H. Barr. Partial-volume bayesian classification of material mixtures in mr volume data using voxel histograms. *IEEE Trans. Med. Imaging*, 17(1):74–86, 1998.

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

[44] Claes Lundström, Patric Ljung, and Anders Ynnerman. Local histograms for design of transfer functions in direct volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 12(6):1570–1579, 2006.

[45] Claes Lundström, Patric Ljung, and Anders Ynnerman. Multi-Dimensional Transfer Function Design Using Sorted Histograms . In Raghu Machiraju and Torsten Möller, editors, *Volume Graphics*, pages 1–8, Boston, Massachusetts, USA, 2006. Eurographics Association.

[46] Claes Lundstrm, Patric Ljung, Anders Persson, and Anders Ynnerman. Uncertainty visualization in medical volume rendering using probabilistic animation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1648–1655, 2007.

[47] P.S. McCormick, J. Inman, J.P. Ahrens, C. Hansen, and G. Roth. Scout: A hardware-accelerated system for quantitatively driven visualization and analysis. In *Proceedings of IEEE Visualization*, pages 171–178, 2004.

[48] K. McDonnell and K. Mueller. Illustrative parallel coordinates. *Computer Graphics Forum (Proceedings of Eurovis 2008)*, 27(3):1031–1038, 2008.

[49] Matej Novotny. Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[50] Alex Pang, Craig M. Wittenbrink, and Suresh K. Lodha. Approaches to uncertainty visualization. *The Visual Computer*, 13(8):370–390, 1997.

[51] Vladimir Pekar, Rafael Wiemker, and Daniel Hempel. Fast detection of meaningful isosurfaces for volume data visualization. In *Proceedings of IEEE Visualization*, pages 223–230, Washington, DC, USA, 2001. IEEE Computer Society.

[52] Wei Peng, Matthew O. Ward, and Elke A. Rundensteiner. Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of IEEE Symposium on Information Visualization*, pages 89–96, Washington, DC, USA, 2004. IEEE Computer Society.

[53] Oleg Polonsky, Giuseppe Patané, Silvia Biasotti, Craig Gotsman, and Michela Spagnuolo. What's in an image?: Towards the computation of the "best" view of an object. *The Visual Computer*, 21(8):840–847, 2005.

[54] Y. Sato, C.-F. Westin, A. Bhalerao, S. Nakajima, N. Shiraga, S. Tamura, and R. Kikinis. Tissue classification based on 3d local intensity structure for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 6(2):160–180, 2000.

[55] Natascha Sauber, Holger Theisel, and Hans-Peter Seidel. Multifield-graphs: An approach to visualizing correlations in multifield scalar data. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):917–924, 2006.

[56] K. Stockinger, J. Shalf, W. Bethel, and K. Wu. Query driven visualization of large data sets. In *Proceedings of IEEE Visualization*, pages 167–174, 2005.

[57] Aleksander Stompel, Eric B. Lum, and Kwan-Liu Ma. Visualization of multidimensional, multivariate volume data using hardware-accelerated non-photorealistic rendering techniques. In *Proceedings of 10th Pacific Graphics Conference on Computer Graphics and Applications (PG '02)*, volume 00, page 394, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

[58] Shigeo Takahashi, Issei Fujishiro, Yuriko Takeshima, and Tomoyuki Nishita. A feature-driven approach to locating optimal viewpoints for volume visualization. *vis*, 00:63, 2005.

[59] Shivaraj Tenginakai, Jinho Lee, and Raghu Machiraju. Salient iso-surface detection with model-independent statistical signatures. In *Proceedings of IEEE Visualization*, pages 231–238, Washington, DC, USA, 2001. IEEE Computer Society.

[60] Ulf Tiede, Thomas Schiemann, and Karl Heinz Höhne. High quality rendering of attributed volume data. In *Proceedings of IEEE Visualization*, pages 255–262, Los Alamitos, CA, USA, 1998. IEEE Computer Society Press.

[61] Fan-Yin Tzeng, E.B. Lum, and Kwan-Liu Ma. A novel interface for higher-dimensional classification of volume data. In *Proceedings of IEEE Visualization*, pages 505–512, 2003.

[62] Jarke Van Wijk and Edward Van Selow. Cluster and calendar based visualization of time series data. In *Proceedings of IEEE Symposium on Information Visualization*, Washington, DC, USA, 1999. IEEE Computer Society.

[63] Chaoli Wang, Hongfeng Yu, and Kwan-Liu Ma. Importance-driven time-varying data visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1547–1554, 2008.

[64] Marc Weber, Marc Alexa, and Wolfgang Müller. Visualizing time-series on spirals. In *Proceedings of IEEE Symposium on Information Visualization*, page 7, Washington, DC, USA, 2001. IEEE Computer Society.

[65] Rainer Wegenkittl, Helwig Löffelmann, and Eduard Gröller. Visualizing the behavior of higher dimensional dynamical systems. In *Proceedings of IEEE Visualization*, pages 119–126, 1997.

[66] Jonathan Woodring and Han-Wei Shen. Multi-variate, time varying, and comparative visualization with contextual cues. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):909–916, 2006.

[67] J. Younesy, T. Moller, and H. Carr. Visualization of time-varying volumetric data using differential time-histogram table. *International Workshop on Volume Graphics*, 0:21–224, 2005.

[68] Hong Zhou, Xiaoru Yuan, Huamin Qu, Weiwei Cui, and Baoquan Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum (Proceedings of Eurovis 2008)*, 27(3), 2008.

Vita

C. Ryan Johnson was born in rural Iowa in October 1980, the son of Lowell and Terry Johnson. He graduated from Carroll High School in 1999 and the University of Northern Iowa with a B.S. Computer Science in 2003. He holds an M.S. Computer Science degree from the University of Tennessee, Knoxville, where he is currently pursuing a Ph.D. Computer Science. He is a lecturer in the Department of Computer Science at Iowa State University.