Pixel Oriented Visualization in XmdvTool

by

Anilkumar Patro

A Thesis

Submitted to the Faculty

of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Master of Science

in

Computer Science

by

August 2004

APPROVED:

Professor Matthew O. Ward, Thesis Advisor

Professor Emmanuel Agu, Thesis Reader

Professor Michael Gennert, Head of Department

Abstract

Many approaches to the visualization of multivariate data have been proposed to date. Pixel oriented techniques map each attribute value of the data to a single colored pixel, theoretically yielding the display of the maximum possible information at a time. A large number of pixel layout methods have been proposed, each of which enables users to perform their visual exploration tasks to varying degrees. Pixel oriented techniques typically maintain the global view of large amounts of data while still preserving the perception of small regions of interest, which makes them particularly interesting for visualizing very large multidimensional data sets. Pixel based methods also provide feedback on the given query by presenting not only the data items fulfilling the query but also the data that approximately fulfill the query.

The goal of this thesis was to extend XmdvTool, a public domain multivariate data visualization package, to incorporate pixel based techniques and to explore their strengths and weaknesses. The main challenge here was to seamlessly apply the interaction and distortion techniques used in other visualization methods within XmdvTool to pixel based methods and investigate the capabilities made possible by fusing the various multivariate visualization techniques.

Acknowledgements

I would like to express my gratitude to my advisor, Prof. Matthew Ward, for his patient guidance and invaluable contributions to this work. I would like to thank Prof. Elke Rundensteiner to make various suggestions for improving the displays. I would also like to thank Prof. Emmanuel Agu for being the reader of this thesis.

I would like to thank my team members, Jing Yang for the clear code handed over to me making my implementation much easier, Nishant Mehta for providing suggestions to my work, Wei Peng and Shiping Huang for making the evaluation process a success, Geraldine Rosario and Punit Doshi for giving help in the beginning of my research.

Thanks also to lots of friends for showing confidence in me.

Last but not least I am grateful to my wonderful parents that are giving me a great moral support.

This work is funded by NSF under grants IIS-0119276.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Our ability to generate and accumulate information has far exceeded our ability to effectively process them. The ubiquitous exchange of information in this technology age is a strong impetus for data growth. The proliferation of the internet today could only hint at the potential of tomorrow's exponential information growth. Advancements in data acquisition technologies results in acquiring data at far greater densities and resolutions than ever before. Such evidence clearly puts forward a case that data size is ever evolving and rising; and that in view of this growth it will be increasingly difficult to process data in search of anomalies, patterns, features, and ultimately knowledge and extrapolation of that knowledge.

Interpreting data, be it overwhelming or not, is an arduous task. But it is most difficult when it is unclear what, in the voluminous data, to look out for. Automated analysis is hopeless when such analysis criteria cannot even be explicitly formulated. This is where visualization plays a crucial and most effective role — by relying upon the power of the trained human eye.

1.1 Multivariate Data Visualization

Multivariate visualization simply refers to the display of multidimensional data. A multidimensional data set consists of a collection of N-tuples, where each entry of an N-tuple is a nominal or ordinal value corresponding to an independent or dependent variable. We distinguish ourselves from the domain-specific class of visualization methods. Our research can be termed as non-domain-specific multivariate visualization, a radically different mode of displaying multidimensional data. Its nondomain specific nature results in generality and as such may be used to display a much larger class of datasets. Examples of such datasets include results from censuses, surveys and simulations. Most analysis of such data to date still relies on the application of statistical computations. Statistical methods, though precise, lack the richness of graphical depictions. But more importantly, they require the user to explicitly define sets of parameters for analysis. This is an arduous task, if not impossible, if the user has little knowledge or intuition about the characteristics of the data that they are about to analyze.

Visualization serves to complement rather than to replace traditional data analysis. Visualization is the graphical presentation of information, with the goal of providing the viewer with a qualitative understanding of the information contents. A visual sense of the data provides an interpretation unprecedented by statistical methods. It facilitates identification of trends and anomalies otherwise missed by statistical analysis due to the difficulty of explicitly formulating analysis parameters. Observations from the visualization session may be further pursued using statistical methods supplied with parameters that befit the visual cues. These visual cues help the user to focus on where to target the quantitative or statistical analysis. Several techniques have been proposed to display non-domain-specific multivariate

data. They have been broadly categorized [Kei00] as:

- *Geometric Projection*: These techniques aim to find interesting geometric transformations and projections of multidimensional data sets. Examples include Scatterplot matrices [And72], Landscapes [Wri95], Hyperslice [WvL93] and Parallel Coordinates [ID90].

- *Icon-based*: The idea here is to map each multidimensional data item to a shape, where data attributes control shape and color attributes. Examples include Chernoff-faces [Che73], Stick figures [PG88], Shape Coding [Bed90], and Color Icons [Lev91].

- *Hierarchical*: These techniques subdivide the k-dimensional space and present the subspaces in a hierarchical fashion. Examples include Dimensional Stacking [WLT96], Worlds-within-worlds [FB90], Treemap [Shn92], and Cone Trees [RMC91].

- *Pixel-based*: Here, each attribute value is represented by one colored pixel and the values for each attribute are presented in separate subwindows. Examples are spiral [KD94] [Kei96], recursive pattern [KKA95], and circle segment techniques [AKK96].

The first three techniques do not scale well with respect to the size of the data set. The main problem with applying such techniques to large data sets is display clutter — that the amount of clutter obscures or occludes any visible trends in the data display. For instance, take the parallel coordinates display in Fig. 1.1. We can easily spot correlations between variables in the data set. However, if we display a larger data set as shown Fig. 1.2, we can hardly discern any relative patterns or anomalies due to the mass of overlapping lines. As a generalization, we

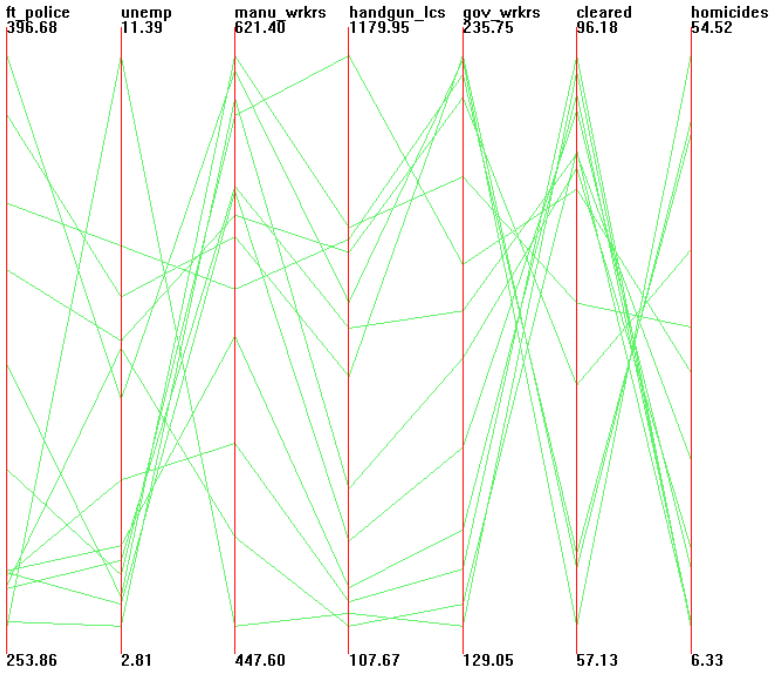

Figure 1.1: Parallel coordinates display of Detroit Homicide data set: a 7-dimensional data set with 13 records. Note the inverse correlations between the number of cleared homicides and both the number of government workers and the total number of homicides.

postulate that any method that displays a single entity per data point invariably results in overlapped elements and a convoluted display that is not suitable for the visualization of large data sets. The quantification of the term "large" varies and is subject to revision in sync with the state of computing power. For our current application, we define a large data set to contain tens of thousands to a million data elements or more. Pixel-based displays are the only set of visualization techniques that aim to effectively visualize such large datasets.

1.2 Pixel Oriented Visualization

Pixel-oriented techniques have been pioneered by Keim for the VisDB system [KD94] as a means for representing large amounts of high-dimensional data with respect to

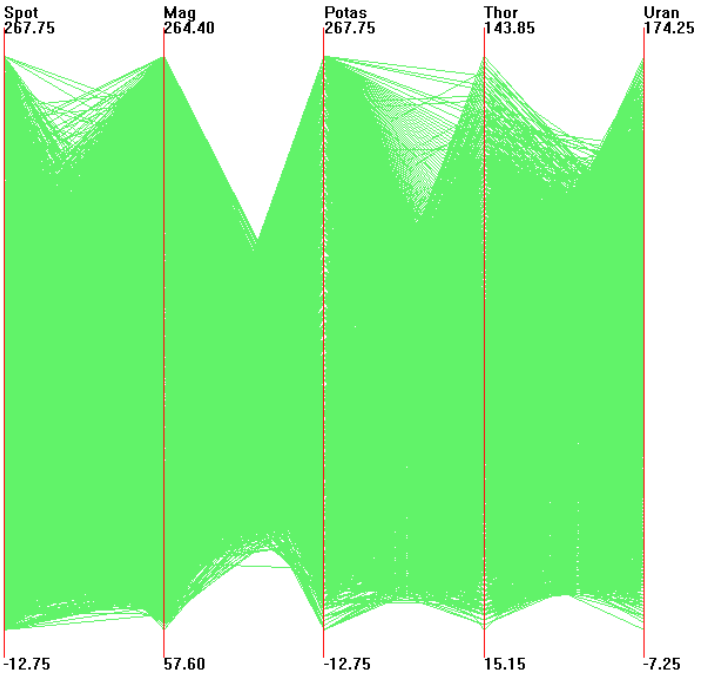

Figure 1.2: Parallel coordinates display of a Remote Sensing data set: a 5-dimensional data set with 16,384 records. Note the amount of over-plotting precludes the perception of any data trends, for instance the relative densities.

a given query. The basic idea of pixel-oriented visualization techniques is to represent each attribute value as a single colored pixel, mapping the range of possible attribute values to a fixed color map and displaying different attributes in different subwindows. Pixel-oriented visualization techniques maximize the amount of information represented at one time without any overlap. They effectively preserve the perception of small regions of interest while still maintaining the global view.

Designing pixel-oriented displays have been discussed at length in [Kei00]. The design considerations are the choice of color space, subwindow shapes, pixel arrangement, dimension ordering and query specification. One of the most important considerations in pixel-oriented displays is the arrangement of the pixels within each of the subwindows. This is important since, due to the density of the pixel displays, only a good arrangement will lead to the discovery of clusters and correlations among

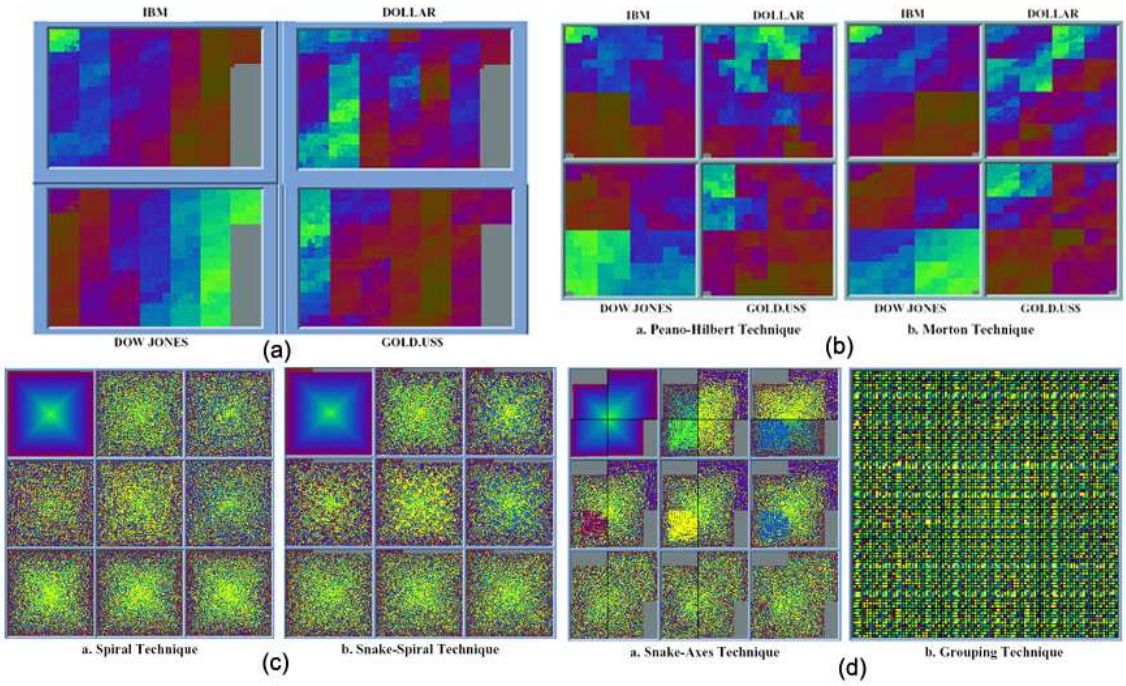

Figure 1.3: Various layouts in Pixel-Oriented Visualization. (a) using Screen-Filling Curve Techniques. (b) using Recursive Pattern Techniques. (c) Query-Dependent using Spiral and Snake- Spiral Techniques. (d) Query-Dependent using Snake-Axes and Grouping Techniques [Images generated in VisDB]

the dimensions. The arrangement defines the layout of the pixels within the display. Query-independent techniques use the natural ordering within the data (for example, time-series data) to define the layout, whereas Query-dependent techniques arrange the pixels in the context of a specific user query and also provide feedback on the query. Figure 1.3 illustrates some of the layout techniques in pixel displays for the visualization of financial data (database contains prices of IBM stock, Dow Jones index, Gold and USD exchange rate from Sep'87 to Feb'95 with 64,800 data values).

Since pixel-oriented visualization techniques maximize the screen space usage, they have been shown [KK95] to be useful for the exploration and analysis of large databases to find interesting data clusters and their properties. They have been proven useful for easily identifying and isolating clusters, correlations and hotspots

in large databases [KK95].

1.3 Open Challenges

Unfortunately, it is difficult to comprehend all information within multivariate data through a single graphical representation, whether it be a parallel coordinates or pixel oriented display. The effectiveness of the visualization can be enhanced by providing the information from multiple views and using multiple techniques within the same environment. Linkage of pixel-oriented visualization techniques with other displays in XmdvTool may in itself provide better understanding of the data. Since pixel-oriented displays are an image-based representation of the data items within a dimension, it would be easy to spot similarity between dimensions and this can be used to validate the results of Dimension Ordering [YPWR03] in XmdvTool. Pixel-oriented techniques can also provide quick query response preview for Subspace Data visualization [Yan03].

By adding interactivity to a visualization system, the understanding of the relationships and patterns within the data can be improved. Due to the complexity and density of the data sets to be analyzed, interactive visualization brings all kinds of potential benefits to exploration by allowing the data to be displayed and manipulated interactively with enough accuracy and speed to be a useful analysis tool. Many interaction techniques have not been clearly defined for current implementations of pixel-oriented displays. Simple interactions such as Zooming, Panning, Distortion, and Selecting (Brushing) have not been addressed in the current implementations and the first step would be to design these interactions, which are quite common in the other displays in XmdvTool. The main challenges here are:

- How to provide direct manipulation of brushes. This might require a focus-

oriented interaction technique to allow the user to confidently and conveniently select the data.

- How to perform data driven brushing.
- How to avoid distortion effects (such as loss of sharpness) when zooming and panning.
- How to keep the context while zooming in different sections of the subwindows. This might require some kind of distortion mechanism (such as fisheye distortions [Fur86]).

Theoretically, pixel-oriented visualizations should maximize the amount of information on the display, but auxiliary displays that are required for some of the interaction mechanisms do take up a lot of space. Overlaying of auxiliary displays on the main visualization might help in maximizing the screen space.

Pixel-oriented techniques are still limited to the number of pixels on the screen. With today's technologies, this means approximately 1024 x 1024 = 1 MB of 1-dimensional or M/N N-dimensional records can be conveniently displayed on the screen. While this number is large, it might not be sufficient for very large databases. Hierarchical Data Visualization techniques [FWR99a] have been incorporated in XmdvTool to address this in other visualization techniques. The combination of hierarchical visualization techniques with pixel-oriented displays has not been studied before. These techniques could identify relationships with the clusters for a particular level of detail in the hierarchy.

1.4 Goals of this Thesis

The purpose of this thesis is to add a new class of visualization techniques, namely pixel-oriented methods, to the already existing repertoire of techniques within XmdvTool. The fact that no effort has been made to date to link pixel-oriented displays with more traditional forms of visualization techniques to gain better insight into the data is the main driving force for this thesis. This brings with itself many challenges such as the design of interaction techniques and the design of displays for hierarchically clustered data within pixel-oriented techniques, similar to those found within other displays in XmdvTool. Hence, the main goal of this thesis will be to study linking between the various visualization types and investigate how interaction and multi-resolution methods in XmdvTool port to pixel-based visualizations.

1.5 Contributions

There are two main contributions of this thesis. They are:

- Integration of pixel-oriented visualization techniques into XmdvTool for linked views with the other displays along with a suite of completely new interaction and distortion techniques for the display.
- Development and implementation of a completely new pixel oriented displays for hierarchically clustered datasets that helps the user to load huge datasets and view relationships among the summarizations.

1.6 Thesis Organization

Recent research work regarding the display of large multivariate data sets are surveyed in Chapter 2. Chapter 3 gives an overview of the XmdvTool system. Chapter

4 describes how to generate a flat pixel oriented display, discusses issues in the display generation and presents the navigation and selection tools for the display. Chapter 5 does the same for hierarchical pixel oriented displays. Chapter 6 gives an overview and a high-level system diagram of our implementation. Chapter 7 presents experiments performed to evaluate the usefulness of linking of pixel oriented displays with other displays. Conclusions and open areas for future work are provided in Chapter 8.

Chapter 2

Related Work

This chapter briefly describes the work done by others in the area of visualizing large multivariate data sets. This chapter also looks into various distortion techniques for keeping overview and detail views of the displays for large data sets.

2.1 Visualization Techniques for Large Multivariate Data Sets

In recent years several research efforts have been directed at the display of large multivariate data sets. One approach is to use compression techniques to reduce the data set size while preserving significant features. For example, Wong and Bergeron [WB96] describe the construction of a multiresolution display using wavelet approximations, where the data size is reduced through repeated merging of neighboring points. Wavelet transforms identify averages and details present at each level of compression. The capability of brushing is also incorporated into their model so that the brushed data is displayed at a different resolution than the nonbrushed data. Figure 2.1 depicts a series of parallel coordinate plots of the same timeseries data

set at different resolutions. Figure 2.1a is the original data. Figure 2.1b presents an overall reduction in resolution. From this diagram, the user localizes or brushes an area of interest (shown by the purple band in Figure 2.1c). Figure 2.1d depicts the brushed area at a higher level of resolution (shown in red). However, wavelet transform requires data to be ordered, making it useful only for data sets with a natural ordering along one dimension, for instance time-series data.

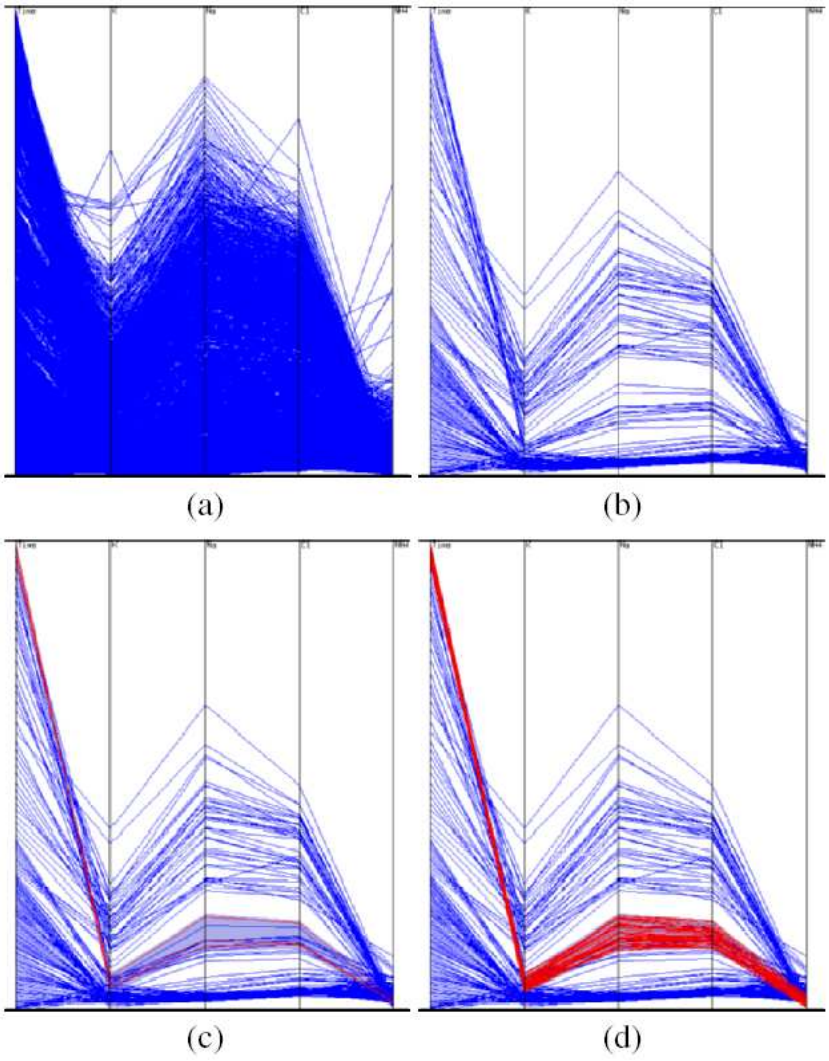

Figure 2.1: Wavelet approximations of a timeseries data set at different resolutions. [Image used from [WB96]]

Another approach is to let the characteristics of the data set reveal itself. For ex-

ample, Wegman and Luo [WL97] suggest overplotting translucent data points/lines so that sparse areas fade away while dense areas appear emphasized. Figure 2.2 shows a parallel coordinates display plotted on a black background. From this plot which looks rather like an x-ray of the internal structure of the data set, high density regions can be identified. The disadvantage of this method is that it relies on overlapping points/lines to identify clusters. Clusters without overlapping elements will not be visually emphasized.

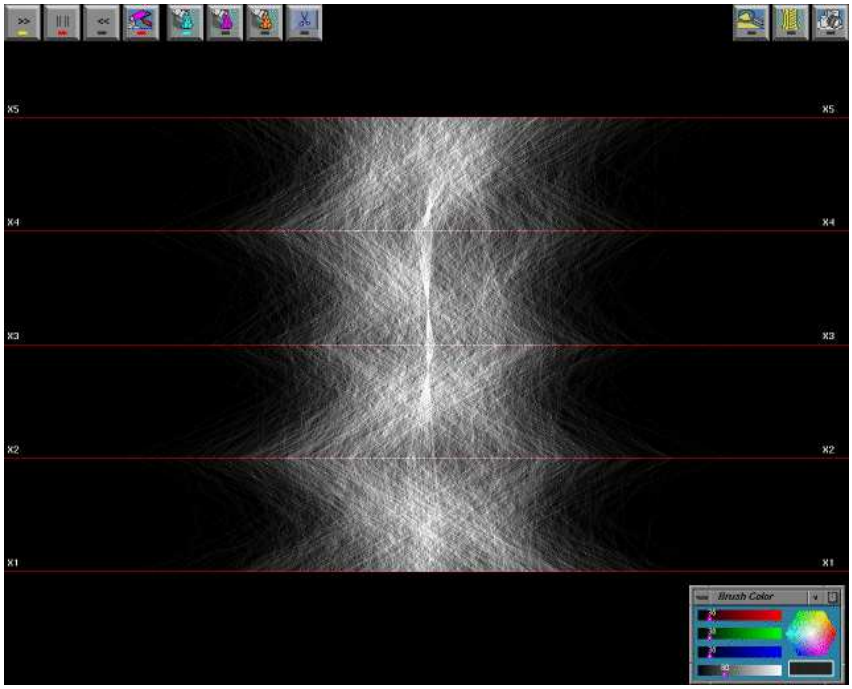

Figure 2.2: Using overplotting to reveal the internal structure of a data set. [Image used from [WL97]]

Keim et al. [Kei96] studied pixel-level visualization schemes that allow the display of large data sets on a typical workstation screen based on recursive layout patterns, where each pixel represents a data element. This enables the user to get a global overview of the data through which he/she can find dimensional correlations and multidimensional clusters. Their visualization system, VisDB, also allows one to focus on displaying as many tuples as possible and to provide feedback as users

refine their queries. This system even displays tuples that do not satisfy the query, indicating their distance from the query criteria using spatial encodings and color. This approach helps the user avoid missing important data points that fall just outside the selected query parameters. This system will be explained extensively in Chapter 4.

Wills [Wil98] describes a visualization technique for hierarchical clusters. His approach expands upon the treemap idea [Shn92] by recursively subdividing the tree based on a similarity measure. Wills used the similarity measure as a value to control the clustering granularity. For instance, with a smaller value, fewer clusters with more elements will be shown. This measure also acts as a level-of-detail control for smooth transitions across treemaps of different granularity. Their main purpose is to display the clustering results, and in particular, the data partitions at a given similarity value. Hence, the N-dimensional characteristics of the clusterings such as the mean or extents information are not displayed along with the treemap. Our research draws on several of the ideas found in the above work.

Fua et al. [FWR99a] have augmented the XmdvTool visualization system with structure-based brushes that allow the user to control the global level of detail (based on hierarchical clustering of data) and to brush records based on their proximity within the hierarchical structure. The data is stored and presented at multiple resolutions using clustering and partitioning techniques. Data aggregation techniques are used to collapse the data into clusters, maintaining the N-dimensional characteristics of the clusterings, and show the population and extents of clusters with bands of varying translucency. However, this approach limits the user, in this case to viewing a single hierarchical structuring of the data and a single ordering of that hierarchy to make proximity meaningful. Details about the hierarchical clustering algorithm and structure based brushing follow in Chapter 5.

Our work relates to both the pixel oriented work done by Keim and hierarchical data visualization by Fua et al. We extend the pixel-oriented techniques suggested and designed by Keim et al. [Kei96, Kei00] by providing better interaction tools, better query-specification mechanisms, subwindow placements to enhance the relation between dimensions, distortion tools and sampling. We have also used hierarchical clustering techniques as found in XmdvTool pioneered by Fua et al. to drive the pixel oriented displays. We remove the limitation of a single ordering for the hierarchy and introduce structure based orderings of the dataset which makes the proximity of the brush meaningful.

2.2 Distortion Techniques

The use of distortion techniques has become increasingly common as a means for visually exploring dense information displays. Distortion operations allow the user to examine a local area of interest in detail and at the same time present a global view of the entire space to provide an overall context to facilitate navigation.

Leung et al. [LA94] provide a taxonomy of distortion-oriented techniques and out-line their underlying relationships. They further classify the techniques into two distinct classes, namely techniques with continuous magnification functions and those without. The FishEye View [Fur86], a well-known distortion technique, belongs to the former class. Distortion techniques such as Bifocal Display [SA82] and Perspective Wall [MRC91] belong to the latter.

The FishEye View concept was first proposed by Furnas [Fur86] as a presentation strategy for information having a hierarchical structure, such as geographical regions or organization charts of companies. The fundamental idea of the technique is thresholding. Each element in the hierarchical structure is assigned a number

based on its importance and another based on its distance from the current point of focus. A threshold value is then selected and compared with a function of these two numbers to determine what information will be presented or suppressed. Consequently, more relevant information will be presented at greater detail whereas less relevant ones are displayed as an abstraction.

In this thesis, we incorporate distortion techniques in our navigation tools to reduce display or information clutter while maintaining context. We introduce a distortion operation similar to FishEye views to allow one to closely examine the details of each subwindow. Weighing factors are also associated with the dimensions to enhance the pixel ordering for finding non-linear correlations among the dimensions.

Chapter 3

Background on XmdvTool

We have added the pixel oriented displays as an extension to the XmdvTool system developed by Ward et al. [War94, MW95, FWR99a, FWR99b, WYR00, YWR02] at Worcester Polytechnic Institute (WPI). We briefly describe the XmdvTool system in this chapter.

3.1 Overview

XmdvTool is a public-domain software package for interactive visual exploration of multivariate data sets. XmdvTool version 6.0 is based on OpenGL and Tck/Tk and is available for Windows95/98/NT/2000/XP and Linux platforms. It already supports four classes of techniques for displaying both (non-hierarchical) flat form data and hierarchically clustered data, namely scatterplots, star glyphs, parallel coordinates, and dimensional stacking. XmdvTool also supports a variety of interaction modes and tools, including brushing in screen, data, and structure spaces, zooming, panning, and distortion techniques, and the masking and reordering of dimensions. Univariate displays and graphical summarizations, via tree-maps and modified Tukey box plots, are also supported. Finally, color themes and user cus-

tomizable color assignments permit tailoring of the aesthetics to the users. XmdvTool has been applied to a wide range of application areas, such as remote sensing, financial, geochemical, census, and simulation data.

We introduce the existing functions of XmdvTool briefly below. To learn detailed information about the flat visualization and N-dimensional brushing techniques, see [War94, MW95]. To learn detailed information about the hierarchical visualization techniques and their interactive tools in XmdvTool, see [FWR99a, FWR99b, WYR00, YWR02]. More information about the XmdvTool project can be obtained from `http://davis.wpi.edu/~xmdv`.

3.2 Flat Visualization Techniques in XmdvTool

3.2.1 Flat Visualization Techniques

XmdvTool supports four methods for displaying flat form data, namely parallel coordinates (see Figure 3.1), star glyphs (see Figure 3.2), scatterplot matrices (see Figure 3.3), and dimensional stacking (see Figure 3.4).

In parallel coordinates [ID90, Weg90], each dimension is represented as a uniformly spaced vertical axis. A data item in this multidimensional space is mapped to a polyline that traverses across all the axes. Figure 3.1 shows the Iris data set (4 dimensions, 150 data items) using parallel coordinates.

In star glyphs, each data item is represented by an individual shape [And72, Che73, RAEM94]. In a star glyph, the data values are mapped to the length of rays emanating from a central point, and the ends of the rays are linked to form a polygon. We can view these rays as axes, with each axis representing a dimension. The directions of these axes are from the center point to the outside. Figure 3.2 shows the Iris data set using star glyphs.

In scatterplot matrices, each data item is projected to $N * N$ plots, with N being the number of dimensions of the data set. The position of the projected point in a plot is decided by the values of the data item in the two dimensions that compose this plot. Figure 3.3 shows the Iris data set using scatterplot matrices.

Flat dimensional stacking [LWW90] displays an N dimensional data set by recursively embedding pairs of dimensions within one another. Each dimension is discretized into a small number of subranges, and two dimensions are initially selected to subdivide the display space into subimages whose size depends on the number of subranges or bins used for those dimensions. These subimages are subdivided based on the next two dimensions, and the process repeats until all dimensions have been mapped. Thus the multivariate space is split into a number of small cells that each map to a segment of the screen. Each data item will fall into one of these small cells, and thus have an assigned screen position. Figure 3.4 shows the Iris data set using dimensional stacking.

3.2.2 Brushing in Flat Visualizations

XmdvTool has implemented N-D brushes in the flat visualizations [MW95] to allow users to select subsets of data items from the visualized data sets. Many useful operations, such as highlighting, magnifying, or saving as new data sets, can be applied to the selected subsets. N-D brushes have the following characteristics:

- N-D hyperbox shape
- step edge or ramp edge boundary
- boolean operations among multiple brushes
- multiple methods to control the sizes and positions of the brushes

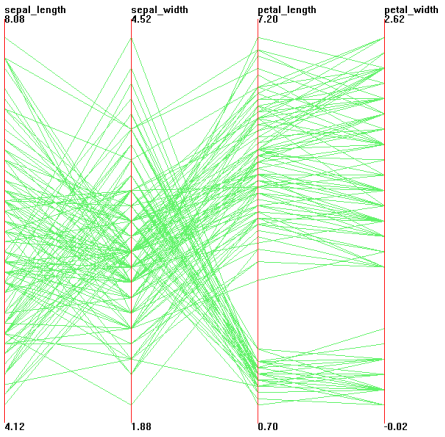

Figure 3.1: Iris Data Set in Flat Parallel Coordinates

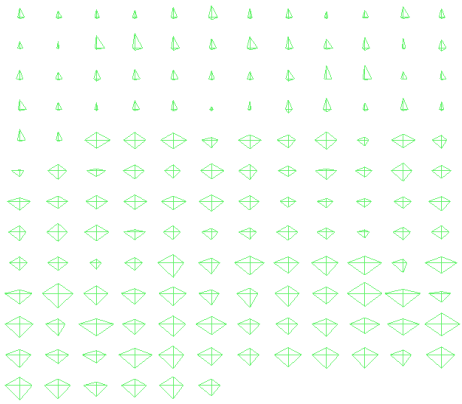

Figure 3.2: Iris Data Set in Flat Star Glyphs

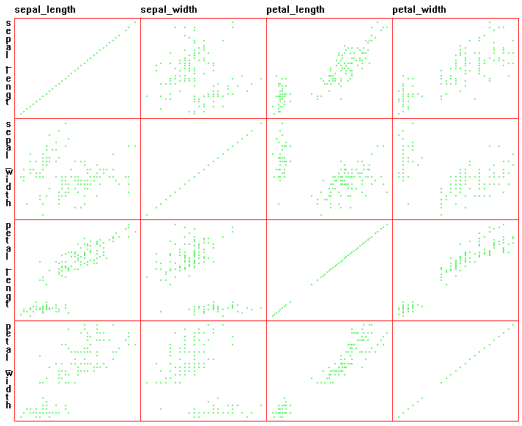

Figure 3.3: Iris Data Set in Flat Scatterplot Matrices

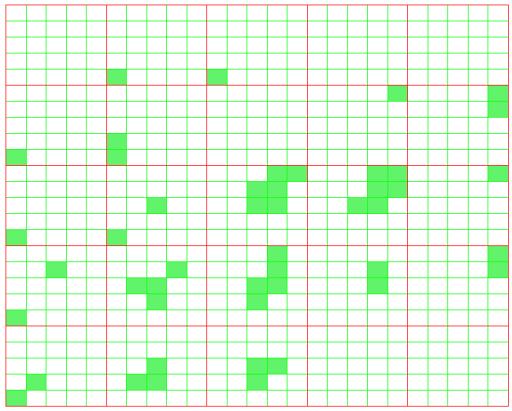

Figure 3.4: Iris Data Set in Flat Dimensional Stacking

XmdvTool provides a function for saving brushed data. This function will save the brushed data as a new file in the XmdvTool format.

3.3 Hierarchical Data Analysis in XmdvTool

3.3.1 Hierarchical Visualization Techniques

The flat visualizations become very crowded when they are applied to large-scale data sets. To overcome this clutter problem, we have developed an Interactive Hierarchical Display framework [YWR02]. The underlying principle of this framework is to develop a multi-resolution view of the data via hierarchical clustering, and to use hierarchical variations of traditional multivariate visualization techniques to convey aggregation information about the resulting clusters. Users can then explore their desired focus regions at different levels of detail, using our suite of navigation and filtering tools.

By applying this framework to the flat visualization techniques, we extend XmdvTool with hierarchical parallel coordinates (see Figure 3.5), hierarchical star glyphs (see Figure 3.6), hierarchical scatterplot matrices (see Figure 3.7) and hierarchical dimensional stacking (see Figure 3.8).

Hierarchical parallel coordinates are an extension of traditional (flat) parallel coordinates. In hierarchical parallel coordinates, the clusters replace the data items. The mean of a cluster is mapped to a polyline traversing across all the axes, with a band around it depicting the extents of the cluster in each dimension. The lower edge of the band intersects each axis at the minimum value of its respective cluster in that dimension. The upper edge of the band intersects each axis at the maximum value of its respective cluster in that dimension. To give the user a sense of the location of data points in a cluster and to convey the overlap among clusters, each band is translucent. We assume that there is a linear drop-off in the density of cluster data from its center to the edge, and set the maximum opacity proportional to the population.

Hierarchical star glyphs are an extension of flat star glyphs. In hierarchical star glyphs, each star glyph represents a cluster. The mean values are used to generate the basic star shape. The band around the mean polygon has two edges; one is outside the mean polygon and another one is inside the mean polygon. The inside edge intersects each axis at the minimum value of its respective cluster in that dimension, while the outside edge intersects each axis at the maximum value of its respective cluster in that dimension. Obviously, if we draw a star glyph starting from the same center point to present a data item included in that cluster, this star glyph would be inside the band of that cluster. Thus the band successfully depicts the extent of the cluster.

Hierarchical scatterplot matrices are an extension of flat scatterplot matrices. The mean of a cluster is drawn as an ordinary data item in flat scatterplot matrices. The extents of each cluster form rectangles around the projected mean in each plot. The projections of a cluster on different plots are drawn in the same color, which gives users the possibility to more easily link a cluster from one plot to another. In the flat form scatterplot matrices, all the data items have the same color. Hence users can have difficulty linking a data item when they move from one plot to another, although selective highlighting helps this linkage.

Hierarchical dimensional stacking is an extension of flat dimensional stacking. The clusters replace the data items. The mean of a cluster will fall into a single small block as if it were an original data item in the flat form dimensional stacking. The band of this cluster depicting the cluster extents may potentially map to many blocks. This time it is possible that some parts of the band are disjoint from others due to the embedding process, even though they are adjacent in N-dimensional space.

3.3.2 Interactive Tools in Hierarchical Visualizations

There are several interactive tools, such as the structure-based brush, drill-down/roll-up operations, extent scaling, and dynamic masking, to help users interactively explore the hierarchical visualizations. These are described briefly below. Details can be found in [FWR99b, FWR00, WYR00, YWR02].

- Structure-based Brush

  A structure-based brush allows users to select subsets of a data structure (for example, a hierarchy) by specifying focal regions as well as a levels-of-detail on a visual representation of the structure.

- Drill-down/Roll-up Operations

  Drill-down/roll-up operations allow users to change the level of detail of interactive hierarchical displays intuitively and directly. Drill-down refers to the process of viewing data at an increased level of detail, while roll-up refers to the process of viewing data with decreasing detail [FWR99a]. The drilling operations are coupled with brushing. XmdvTool permits selective drill-down/roll-up of the brushed and non-brushed region independently.

- Extent Scaling

  Extent scaling solves the problem of overlapping among the bands by decreasing the extents of all the bands in each dimension by scaling them uniformly via a dynamically controlled extent scaling parameter. Users can still differentiate between clusters with large and small extents after the bands have been scaled.

- Dynamic Masking

Dynamic masking refers to the capability of controlling the relative opacity between brushed and unbrushed clusters. It allows users to deemphasize or even eliminate brushed or unbrushed clusters. With dynamic masking, the viewer can interactively fade out the visual representation of the unbrushed clusters, thereby obtaining a clearer view of the brushed clusters while maintaining the context of unbrushed areas. Conversely, the bands of the brushed clusters can be faded out, thus obtaining a clearer view of the unbrushed region. Used together with the structure-based brush, dynamic masking reduces the overlapping and density of the clusters on the screen by fading out uninteresting clusters so that users can concentrate on the clusters of interest.

3.4 Common Interactive Tools Used in Both Flat and Hierarchical Visualizations

There are some interactive tools in XmdvTool that can be used in both the flat and hierarchical displays. There are briefly described below:

- Zooming and Panning

  The whole display area can be zoomed and panned.

- Dimension Distortion

  In Parallel Coordinates, the distance between two adjacent axes can be increased or decreased. In Scatterplot Matrices, the size of a single plot can be enlarged or reduced.

- Dimension Enabling/Disabling and Reordering

  Dimensions can be enabled/disabled and reordered.

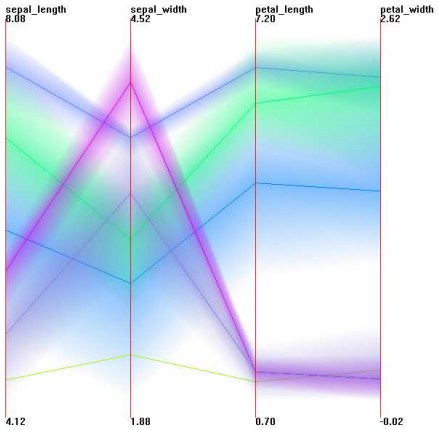

Figure 3.5: Iris Data Set in Hierarchical Parallel Coordinates

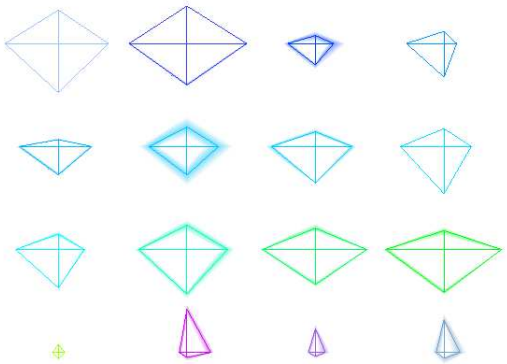

Figure 3.6: Iris Data Set in Hierarchical Star Glyphs

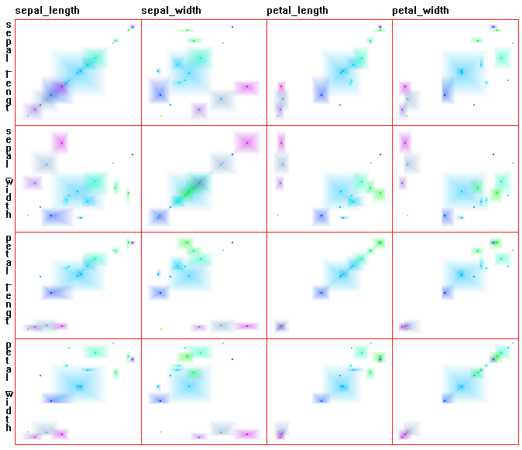

Figure 3.7: Iris Data Set in Hierarchical Scatterplot Matrices

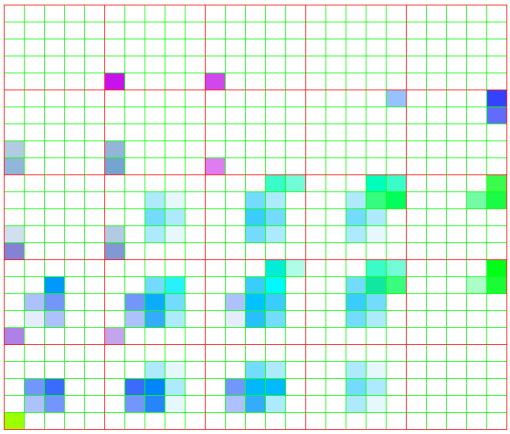

Figure 3.8: Iris Data Set in Hierarchical Dimensional Stacking

Chapter 4

Flat Pixel Oriented Visualization

Two aspects distinguish pixel oriented visualization techniques from other techniques used in XmdvTool. First, the number of data items that can be visualized without overlap is much higher than in other approaches. In general, pixel oriented techniques use each pixel of the display to represent one data value. This means that the number of data values that can be visualized at one point of time is only limited by the number of pixels on the display. A second unique feature of the technique is that the generated visualizations can be query-dependent. Query-dependency means that not only the data items fulfilling the query are visualized, but also a number of data items that approximately fulfill the query. This chapter introduces these two aspects of pixel oriented visualization techniques and discusses the implementation in XmdvTool.

4.1 Introduction

In very large databases with tens of thousands or even millions of data items, it is often a problem to find the data in which a person is interested. Scientific, engineering, and environmental databases, for example, contain large amounts of data

that in many cases are collected automatically via sensors and (satellite) monitoring systems. In querying such systems, even users who are experienced in using a database and query system may have difficulties finding the interesting data spots. If the user does not know the data and its distribution exactly, many queries may be needed to find the interesting data sets. The core of the problem in searching huge amounts of data is the process of query specification. With todays database systems and their query interfaces, a person has to issue queries in a one-by-one fashion. Generally, there are no possibilities to slightly change a query or to express vague queries. Most importantly, the user gets no feedback on his query, except the resulting data set that may contain either no data items and thus no hint for continuing the search, or too many data items and thus too many to look at.

Many approaches have been made to improve the database query interface by providing better feedback in cases of unexpected results. One approach consists of graphical database interfaces that allow the user to browse the data (e.g. FLEX [Mot90] or GRADI [KL92]). Another approach uses cooperative database interfaces [Kap82] that try to give approximate answers in cases where the queries do not provide a satisfactory answer. Such systems use techniques such as query generalization that involves dropping or relaxing a dimension in cases where the original queries fail, and statistical approximations or intensional responses instead of full enumeration in the case of large results (the key ideas were presented for the first time in [JKL77]). Cooperative systems mainly help the user to understand the results and to refine erroneous queries, but do not help to find interesting properties of the data such as functional dependencies, local correlations, or exceptional data items.

Another area that relates to our work is the area of information retrieval. In information retrieval, a lot of research has been done to improve recall and precision in querying databases of unstructured data, such as (full) text. User-provided rele-

vance assessments of results may be used to re-rank the results or to re-run adapted queries [SM83].

4.2 Visualizing Large Data Sets of Multidimensional Data

Many approaches to visualize arbitrary multivariate, multidimensional data have been proposed for various purposes in different application contexts. Many examples can be found in the well-known books of Bertin [BSB82] and Tufte [Tuf82]. More recent techniques include shape coding [Bed90], worlds within worlds [FB90], parallel coordinates [ID90], stick figures [PG88], dimensional stacking [WLT96], and dynamic methods as presented in [MZ92]. In dealing with data sets consisting of tens of thousands to millions of data items, the goal is to visualize as many data items as possible at the same time to give the user some kind of feedback on the query. The obvious limit for any kind of visualization is the resolution of current displays, which is on the order of one to three million pixels, e.g. in the case of our 19 inch displays with a resolution of 1280 x 1024 pixels about 1.3 million pixels. Important is the interactiveness of such a system, which suggests that a new querying system is required. Empirical studies show that interactive, slider- based interfaces improve efficiency and accuracy in accessing databases considerably [Shn94]. Equally important is the possibility of getting immediate feedback on the modified query. By playing with such a system, the user may learn more about the data than by issuing hundreds of queries.

4.3 A New Query Paradigm

In todays database systems, queries are specified in a one-by-one fashion. This is adequate if the user of the database exactly specifies the desired data and accesses a clearly separated data set. For many application areas where databases are used on a regular basis, e.g. accounting, reservation systems, and so on, queries are often based on keys, accessing exactly the desired data. In other application areas, however, especially those with very large data volumes such as scientific, engineering and environmental databases, it is often difficult to find the desired data. If, for example, researchers in environmental science are searching a huge database of test series for significant values, they might be looking for some correlation between multiple dimensions for some specific period of time and some geographic region. Since none of the parameters for the query are fixed, it is in general very difficult to find the desired information. With a query interface that provides users more feedback on the results of their queries, the researchers would probably start to specify one query that corresponds to some assumption and after issuing many refined queries and applying statistical methods to the results, they might find some interesting correlation.

4.3.1 Query Specification in XmdvTool

Query Specification in XmdvTool is through N-dimensional brushing. *Brushing* is the process of interactively painting over a subregion of the data display using a mouse, stylus, or other input device that enables the specification of location attributes. It has been used as a method for performing selection in graphics for many years. The principles of brushing were first explored by Becker and Cleveland [BC87] and applied to high dimensional scatterplots. In this system, the user

specified a rectangular region in one of the 2D scatterplot projections and based on the mode of operation, points in other views corresponding to those falling within the brush were highlighted, deleted or labelled. Ward and Martin [War94, MW95] extended brushing to permit brushes to have the same dimensionality as the data (N-D instead of 2-D). The goal is to allow the user to gain some understanding of the spatial relationships in N-space by highlighting all data points that fall within a user-defined, relocatable subspace.

One common method of classifying brushing techniques is by identifying space in which the selection is being performed, namely screen or data space. This can then be used to specify a *containment criteria* (whether a particular point is inside or outside the brush). In *screen space* techniques, a brush is completely specified by a 2-D contiguous subspace on the screen. In *data space* techniques, a complete specification consists of either an enumeration of the data elements contained within the brush or the N-D boundaries of a hyperbox that encapsulates the selection. A third category, namely *structure space* techniques, allows selection based on structural relationships between data points as discussed in Chapter 5.

In addition, brush manipulation may be *direct* or *indirect*. Direct manipulation refers to the ability to interactively control brush creation and manipulation by mouse (or other locator) actions on the data display itself. On the other hand, indirect manipulation refers to the use of separate widgets such as sliders to specify or manipulate the brush coverage. Direct manipulation is generally preferred for data-driven operations (such as isolating an interesting subset of the display), while user-driven operations (such as a range query) are often easier to specify with indirect methods.

The brush region provides the query for the data to be selected. Various brush operations are defined in XmdvTool that can be performed on the data selected by

the brush, important among them being *highlighting* and *linking*. Highlighting is one of the most fundamental brush operations. Points that are contained by the brush are colored differently from other points to make them stand out. Linking is the ability to select data in one display and see the same data selected in another display. This is useful when multiple methods of visualization are being used in conjunction, as is the case in XmdvTool. By default, XmdvTool provides brush linking between all views.

4.3.2 Query Specification in Pixel Oriented Displays

Brushes in XmdvTool correspond to N-dimensional range queries for databases. The data objects that are within the query region form the result of the query. In most cases, the number of results cannot be determined a priori; the resulting data set may be quite large, or it may even be empty. The results are visually represented in other displays of XmdvTool as differently colored points to make them stand out. This would correspond to large number of red colored polylines in the parallel coordinate display for large results and to none for empty results. To give the user more feedback, not only the data objects that are within the query region are presented in pixel oriented display, but also those that are "close" to the query region and only approximately fulfill the query. The approximate results are determined by using distance functions for each of the dimensions which are combined into the overall distance. The distance functions are datatype and application dependent and must be provided by the application. Examples of distance functions are the numerical difference (for metric types), distance matrices (for ordinal and nominal types), lexicographical, character- wise, substring or phonetic difference (for strings), and so on. The distance function yields distance tuples $(d_i^1, d_i^2, d_i^3, ..., d_i^k)$ that denote the distances of the data object a_i to the query. Having calculated the distances for

each of the dimensions, the distances are combined into the overall distance d_i^{k+1} to the query region. The overall distance d_i^{k+1} would be close to 0 for objects that are within the query region and increasing for the other objects.

The combination of distances for the different dimensions can be a problem because the distances for the different dimensions now have to be considered with respect to the distances of the other dimensions and the combined distance must be defined and meaningful globally. One problem is that the values calculated by the distance functions may be in completely different orders of magnitude. A second problem is that the relative importance of the multiple dimensions is highly user and query dependent. The first problem can be solved by a normalization of the distances. A simple normalization may be defined as a linear transformation of the range $[0, d_{max}]$ (or $[d_{min}, d_{max}]$ in case of bi-directional distances) for each dimension to a fixed range (e.g. [0, 255]). The second problem can only be solved by user interaction since only the user is able to determine the priority of the dimensions. Therefore, in general, it is necessary to obtain weighting factors $(w^1, w^2, w^3, ..., w^{k+1})$ representing the order of importance of the dimensions assigned by the user. This is somehow similar to techniques used in information retrieval that have been proposed to allow a ranking of the resulting data according to its relevance for the query.

With this kind of pixel oriented display in the XmdvTool system, the query specification process would be much easier. In the beginning, the users will see the default N-dimensional range query where the extents of each dimension is set to the mid 50% of its complete range. Then, guided by the visual feedback, they may interactively change the query according to their impression of the visualized results. In exploring very large databases, the visualization of results coupled with the possibility to incrementally refine the query are an effective way of finding the interesting properties of the data. The key idea of the system is to use the phenom-

enal abilities of the human vision system, which is able to analyze small to mid size amounts of data very efficiently and recognizes immediately patterns in images that would be very difficult (in some cases even impossible) if done by a computer.

4.4 Flat Pixel Oriented Implementation

The basic idea of pixel-oriented visualization techniques is to use each pixel of the screen to visualize the data items relevant in the context of a specific query. Each distance to the query is mapped to a colored pixel and these values belonging to one dimension are presented in a separate subwindow. Pixel-oriented techniques partition the screen to m subwindows for data sets with m dimensions. An additional $(m+1)$th subwindow is provided for the overall distance. Inside the subwindows, the data values are arranged according to their relevance with respect to the query. The sorting is necessary to avoid completely sprinkled images that would not help the user in understanding the data. Correlations, functional dependencies and other interesting relationships between dimensions may be detected by relating corresponding regions in the multiple windows.

To achieve that objective a number of design issues have to be examined. The issues can be categorized into the static display part and the dynamic interaction part. The display issues to be considered are:

- How to map the values to colors? A good mapping is obviously very important, but has to be carefully engineered to be intuitive.
- How to arrange the pixels within the subwindow? This can be described formally as an optimization problem.
- What shapes to use for the subwindows? This depends on the number of data

points and dimensions.

- How to order the subwindows for the dimensions? This is very important because dependencies and correlations between dimensions might best be detected if they are placed next to each other.

- Where to place the subwindows on the screen? The positions of the subwindows might reveal more relationships between dimensions.

The issues to be considered for the interaction mechanism are:

- How to interactively specify the query or the brush region? A good interface will be able to improve the speed and accuracy of getting the results.

- Can distortion be used to enhance the visualized patterns and/or improve the query specification?

4.4.1 Display Issues

Color Mapping

Visualizing the distance values using color corresponds to the task of mapping a color scale to a single parameter distribution. The advantage of color over gray scales is that the number of just noticeable differences (JNDs) is much higher [HL92]. The main task is to find a path through color space that maximizes the number of JNDs, but at the same time, is intuitive for the application domain. From a perceptual point of view, brightness is the most important characteristic for distinguishing colors corresponding to a single parameter distribution [HL92]. Hence, it is sufficient to use a color scale with a monotonically increasing (or decreasing) brightness while using the full color range.

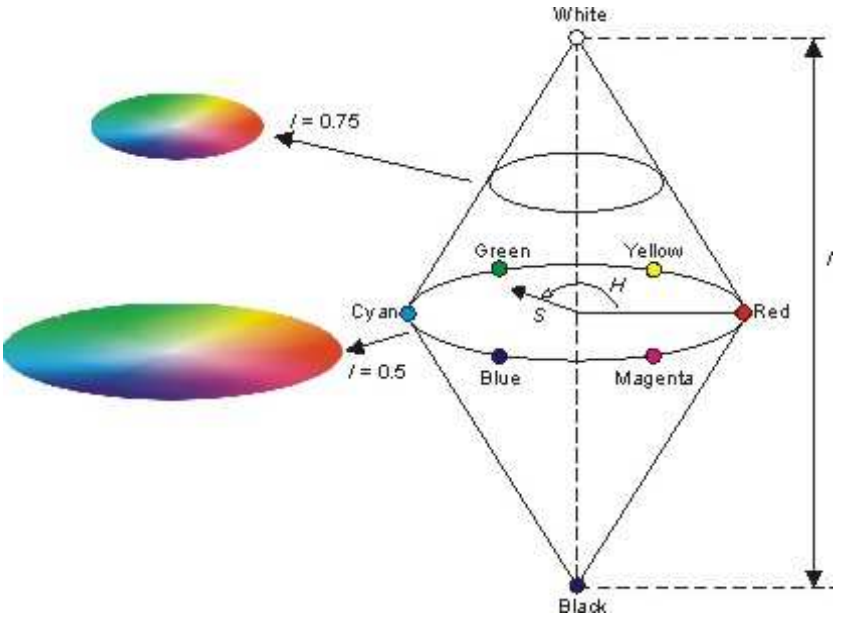

Figure 4.1: HSI Color model used for the color mapping in pixel oriented displays.

Keim et al. have experimented with different colormaps varying in hue, saturation and value. The experiments revealed that the mapping of colors has a high impact on the intuitiveness of the system. The user, for example, may implicitly connect good answers with light colors and bad answers with dark colors or the user may be accustomed to green colors for good answers and red colors for bad answers (like the colors used for traffic lights). They concluded that the parameters of the color mapping should therefore use a monotonically decreasing brightness (intensity, lightness or value), a color ranging over the full scale (hue), and a constant (full) saturation. They have experimentally defined a colormap with the hue (color) ranging from yellow over green, blue and red to almost black is a good choice to denote the distance from the query. The HSI (Hue, Saturation and Intensity) color model (Figure 4.1) provides a color scale with monotonically decreasing brightness. Here, colors with constant intensity and saturation form a circle and linear interpolation provides color scales whose lightness ranges continuously from light to dark. The exact mathematical definition of the HSI parameters in terms of the RGB (Red,

Green and Blue) components is as follows:

$$mid = (\frac{r+g+b}{3})$$

$$intensity = mid + \sqrt{\frac{2}{3} \times (r^2 + g^2 + b^2)}$$

$$saturation = \frac{2 \times (intensity - mid)}{intensity}$$

$$hue = \arccos(\frac{(2 \times r - g - b)}{\sqrt{6} \times \sqrt{r^2 + g^2 + b^2}})$$

The algorithms for generating HSI color values and converting HSI to RGB and vice versa are provided in [KK95]. The parameters for generating the color scales - including the HSI color scale used for the visualization presented in the rest of the thesis - are shown in Table 4.1.

	hue	saturation	intensity
HSI_{min}	1.5 (= Light Green)	1.0	0.4
HSI_{max}	1.0 (= Yellow)	1.0	1.0

Table 4.1: Parameters for generating color scales

The usefulness of the colormaps varies depending on the the user and the application. The user may want to define different colormaps and use them instead of the standard colormap. We have provided a Colormap Editor in XmdvTool to allow the user to define colormaps. Figure 4.2 shows a screen dump of the Colormap Editor in XmdvTool. The user can define the colormap not only for the unbrushed region, but also the brushed region, which might indicate the usefulness of the query. The user can specify the color model (RGB, HSI or Greyscale) to use for the color map through the Editor. The Editor also allows one to define non-linear ramps for all the color models.

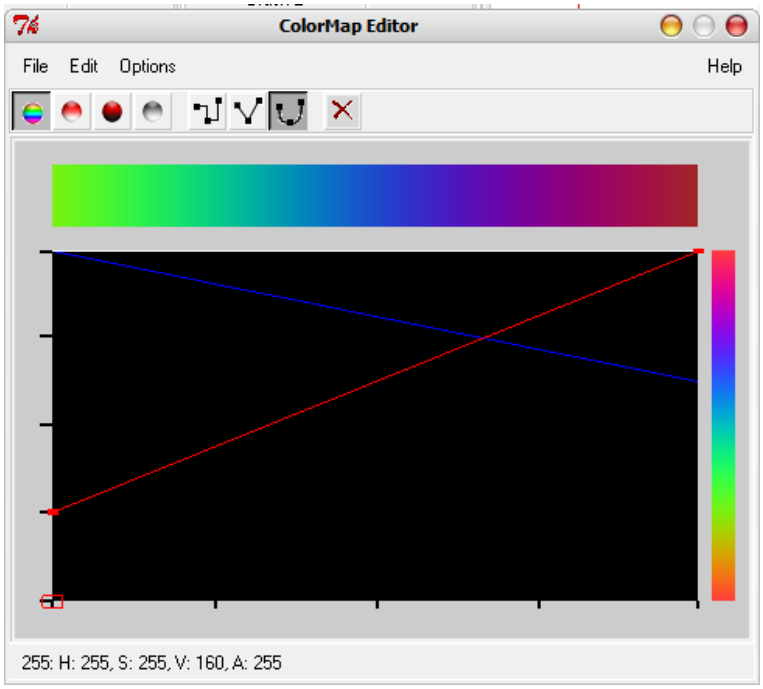

Figure 4.2: Colormap Editor for pixel oriented displays in XmdvTool.

Subwindow Shapes

To relate the visualization of the overall result to visualizations of the different dimensions, we generate a separate window for each dimension. In the separate windows we place the pixels for each data item at the same relative position as the the data item in the overall result window. All the windows together make up the multidimensional visualization. By relating corresponding regions in the different windows, the user is able to perceive data characteristics such as multidimensional clusters or correlations. Additionally, the separate windows for each of the dimension provide important feedback to the user, e.g. on the restrictiveness of each of the dimension and on single exceptional data items.

The rectangular shape of the subwindows allows good screen usage, but at the same time leads to a dispersal of the pixels belonging to one data object over the whole screen. This is especially true for data sets with many dimensions, which

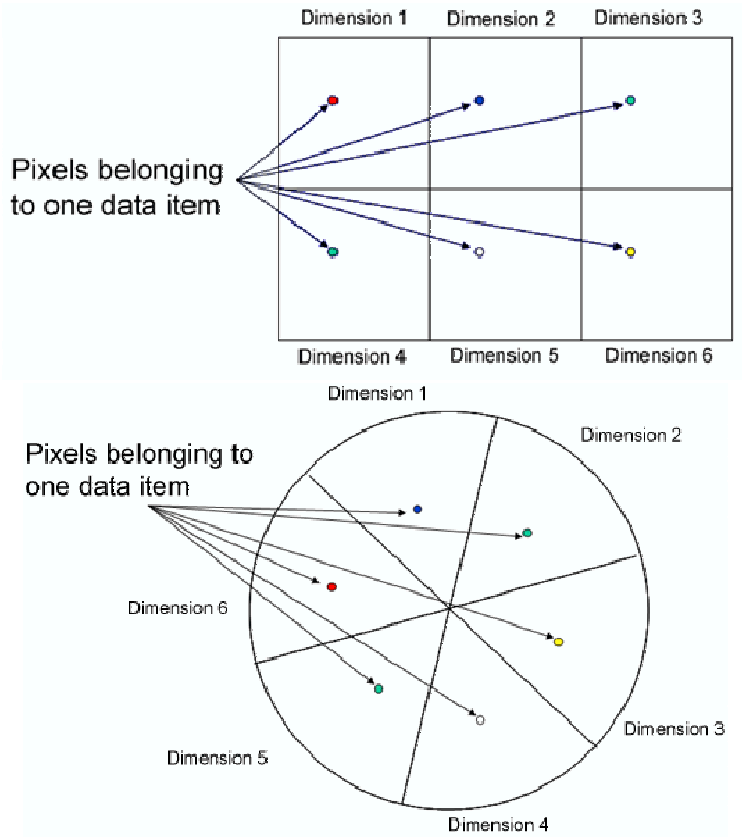

Figure 4.3: Rectangular and Circular shapes of subwindows.

makes it difficult to detect any patterns. Keim et al have suggested using a circle segments technique (refer Figure 4.3) to optimize the distance between the pixels belonging to one data object. The fundamental idea is to display the data dimensions as segments of a circle. If the data consists of k dimensions, the circle is partitioned to $k+1$ segments, the extra one being for the overall distance. The main advantage of this technique is that the overall representation of the whole data set is better perceivable - including potential dependencies, analogies and correlations between dimensions.

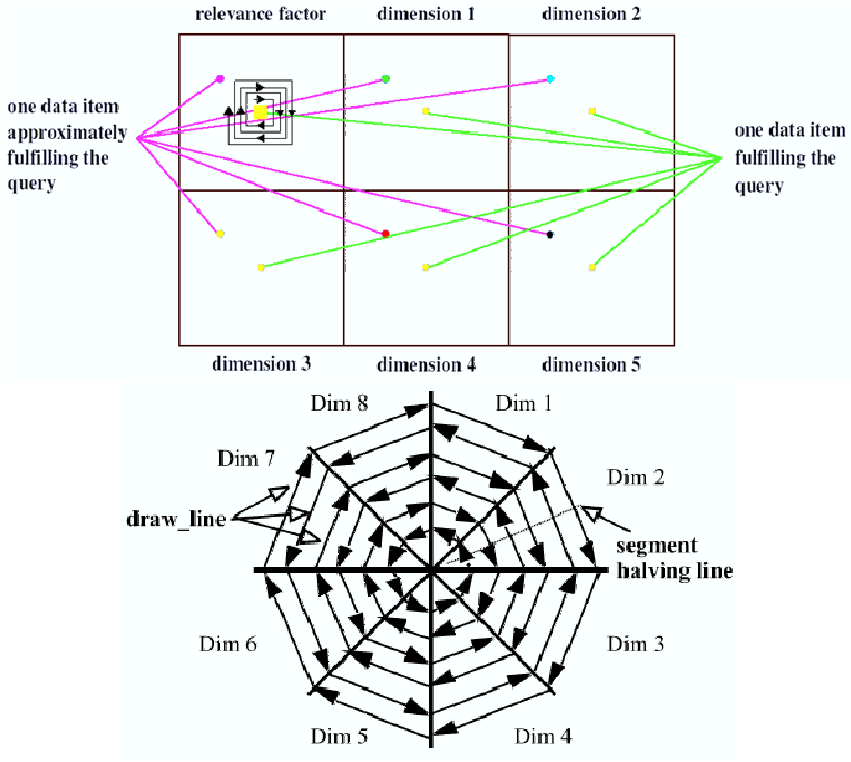

Figure 4.4: Rectangular and Circular segment arrangement of pixels.

Arrangement of Pixels

An important question in designing the system was how to arrange the values displayed on the screen. This is important since, due to the density of the pixel displays, only a good arrangement will allow discovery of clusters and correlations among the dimensions. Keim et al [KD94] tried several arrangements such as top-down, left-to-right, centered and found that arrangements with the lowest overall distances centered in the middle of the window seemed to be the most natural ones. The one closest to the query are colored yellow in the middle and the approximate answers create a rectangular spiral around this region (see Figure 4.4).

For rectangular subwindows, this is derived as a solution to the optimization problem of arranging the pixels in [Kei00]. The pixel arrangement problem is the problem of finding a mapping of the data objects $\{a_1^k, a_2^k, ..., a_n^k\}$ to a subwindow of

size $(w \times h)$, i.e., a bijective mapping $f : \{1 \ldots n\} \rightarrow \{1 \ldots w\} \times \{1 \ldots h\}$ such that

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \left| d(f(i), f(j)) - d((0,0), (w \cdot \sqrt{\frac{|i-j|}{n}}, h \cdot \sqrt{\frac{|i-j|}{n}}) \right|$$

is minimal where $d(f(i), f(j))$ is the L^P-distance of the pixels belonging to a_i and a_j, and

$$\sum_{i=1}^{n} \left| d(f(i), (\frac{w}{2}, \frac{h}{2})) - d((0,0), (\frac{w}{2} \cdot \sqrt{\frac{i}{n}}, \frac{h}{2} \cdot \sqrt{\frac{i}{n}})) \right|$$

is minimal where $d(f(i), (\frac{w}{2}, \frac{h}{2}))$ is the L^P-distance of the pixels belonging to a_i from the center. The first condition in both the definitions aims at preserving the distance of the one-dimension ordering in the two-dimensional arrangement as much as possible. The second portion adds the constraint that the distance to the center should correspond to the overall distance as much as possible.

For circular segment subwindows, the data objects within one segment can be arranged in a back and forth manner along the so-called "draw-line", which is orthogonal to the line that halves the two border lines of the segment (see Figure 4.4).

Ordering of Dimensions

Ordering of dimensions is a general problem that arises for a number of other techniques, such as parallel coordinates and dimensional stacking. The basic problem is that the data dimensions have to be positioned in some one- or two- dimensional ordering on the screen and this is usually done more or less by chance - namely, the order in which the dimensions happen to appear in the data set. The ordering of the dimensions, however, has a major impact on the expressiveness of the visualization, be it parallel coordinates or pixel oriented displays. Parallel coordinates and the circle segments technique require a one-dimensional ordering of the dimen-

sions. In case of the spiral pixel oriented display, a two-dimensional ordering of the dimensions is required.

XmdvTool provides an interactive hierarchical dimension ordering tool that allows for Similarity-based, Importance-based and Interactive dimension reordering. Pixel oriented displays reflect the ordering selected by the automatic tools or set manually.

Subwindow Placement

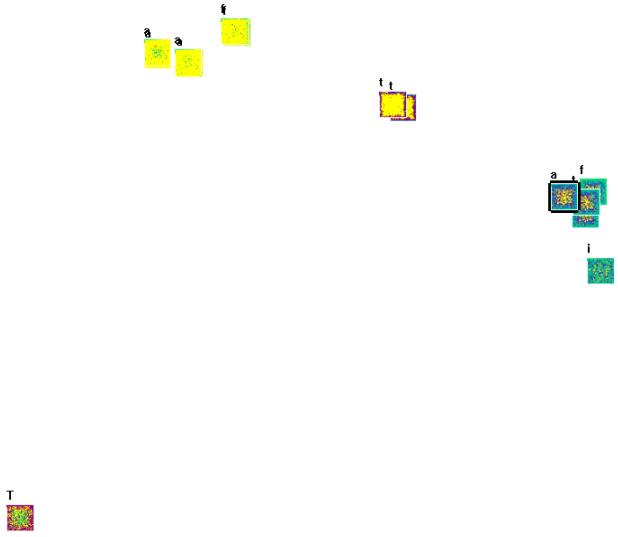

Figure 4.5: Subwindow placement by MDS algorithm for AAUP dataset.

The pixel oriented techniques as defined by Keim [Kei00] do not suggest any particular placement for the subwindows. The subwindows are arranged in a raster-like fashion on the screen, but the subwindows can be placed such that the placement conveys more information about the relationships among the dimensions. The positions of the subwindows can be generated using Multidimensional Scaling (MDS) [KW78] according to the pair-wise relationships among the dimensions. MDS is a technique that maps locations in high dimensional space to positions in a low dimensional space. It is widely used in visualization applications to convey relationships

among data items within a multi-dimensional dataset. For example, [WTPea95] used MDS to map data items in a document dataset to a 2D space and generated a Galaxies display as a spatial representation of relationships within the document collection. The pixel oriented display uses MDS in a different way in that it maps dimensions rather than data items in a dataset to a 2D space according to relationships among the dimensions. Thus, closely related dimensions have positions adjacent to each other.

To get a good set of positions, we first need to identify the factors that affect the subwindow positions. A distance matrix records the correlation of the distances of the data points with respect the user-specified query between each pair of dimensions in the dataset. There are many different correlation measures [ABK98]. We get the similarity distance matrix for the dataset based on the Euclidean distance and use the positions generated by MDS based on this correlation measure. The positions reveal more insights into the data because the positions of the subwindows change when the query (brush) extents are modified. This allows one to observe the similarity between dimensions for a specific range of the brush and accurately determine the range for which two dimensions seem to be strongly correlated. The animation resulting from modification of the brush also gives a better picture of the relationships.

4.4.2 Interaction and Navigation tools

A rich set of navigation and selection tools has been developed for the pixel oriented display in XmdvTool. Zooming and panning, brushing, distortion and pixel reordering help users learn information about the dataset. Manual selection tools allow users to perform human-driven dimension reduction by selecting subsets of dimensions for further exploration using the display as well as other multi-dimensional

visualization techniques.

Brushing

As discussed in section 4.2, brushing in XmdvTool has been categorized either as brushing in screen-space, data-space or structure-space. Both, screen-space and data-space, types of brushing have been implemented with a slight variation on the usage pattern when compared to brushing interactions in other displays. Screen-space brushing is accomplished by modifying the brush extents on a separate colormap display for the currently selected dimension shown within the same space ((see Figure 4.6) or an auxiliary dialog showing colormaps for all dimensions (see Figure 4.7).

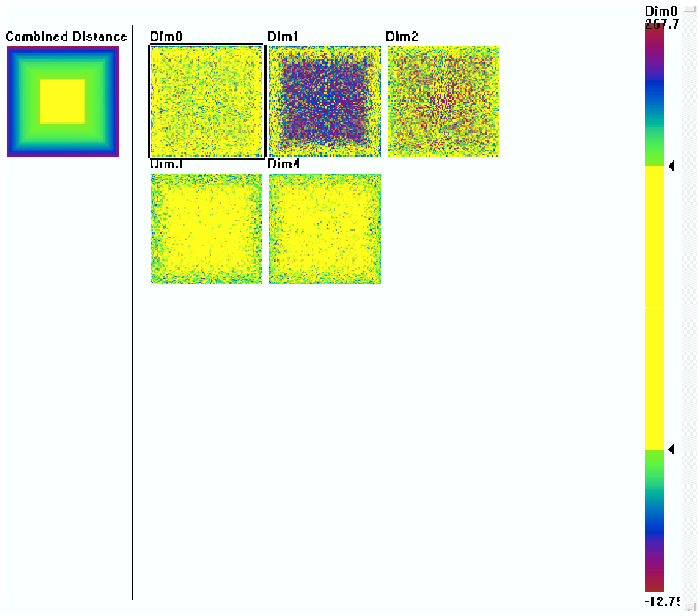

Figure 4.6: Brush region can be changed by moving the markers on the colormap.

The user can change the extents by dragging the brush handles or translate the entire brush to select different extents but keep the same size. The brush can also be incrementally reduced or increased in size by a middle-click of the mouse on

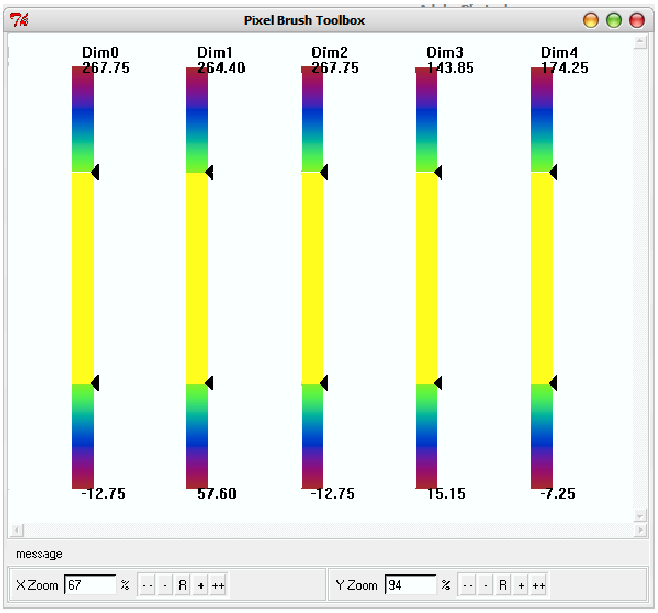

Figure 4.7: Brush region can be changed through the auxiliary brush toolbox.

the brushed section and translation of the mouse to indicate the direction. Data-space brushing is accomplished by manually performing a paint-like operation on one of the subwindows. This selects all the data points that have been painted over (see Figure 4.8). This operation can be slightly confusing if the user paints over a dimension that has not been sorted, but nevertheless is a very useful technique and complements the data-space brushing found in other displays.

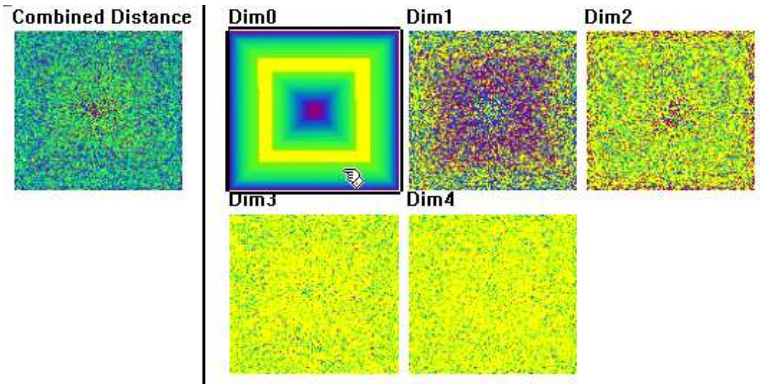

Figure 4.8: Data-space brushing is accomplished by Shift+Mouse clicking and painting over the subwindow. Note that the mouse cursor changes to a hand icon when brushing in data-space.

Distortion

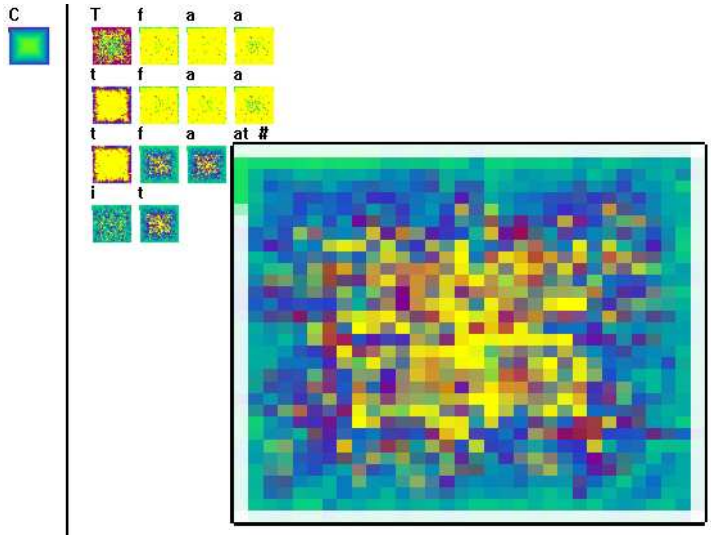

Figure 4.9: Distorted dimension for the AAUP dataset.

Users can interactively enlarge the size of some subwindows while keeping the size of all the other subwindows fixed. This is accomplished by using the distortion slider in a helper GUI. In this way users are allowed to examine details of textures of the enlarged subwindows within the context provided by the other subwindows. Figure 4.9 gives an example of distortion.

Zooming and Panning

Users can zoom in, zoom out and pan the pixel oriented display using the corresponding controls provided in XmdvTool. For example, in order to reduce overlaps, sometimes the size of the subwindows has to be set very small when there are a large number of dimensions. Zooming into the display will enlarge the subwindows so that the user can have a clear view of the texture of the subwindows.

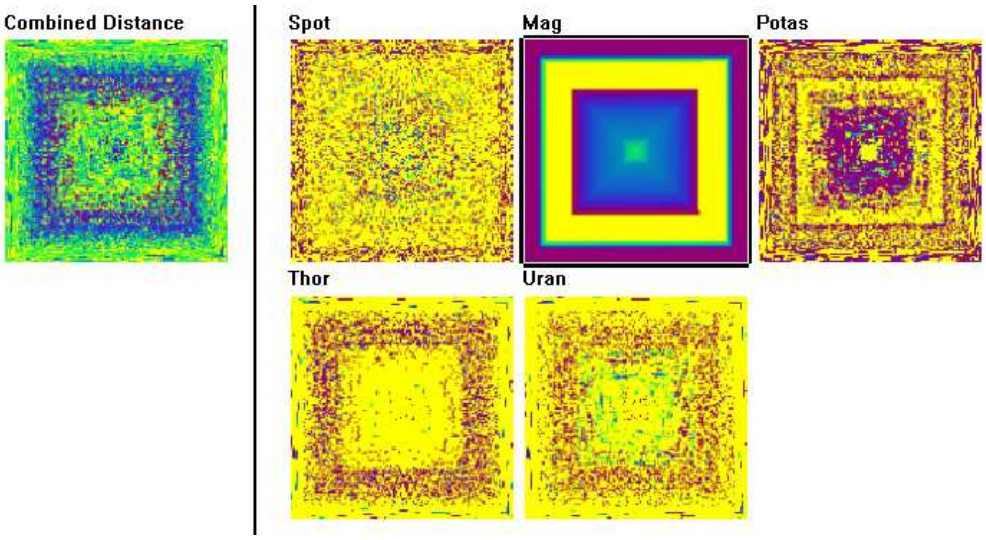

Figure 4.10: Manual Pixel Reordering sorted on Magnetics dimension on the Remote Sensing Dataset.

Manual Pixel Reordering

We allow users to select a dimension on which the data items are sorted by selecting the subwindow and then clicking the middle button on the mouse. Subwindows will have different textures, thus different patterns of the dataset will be revealed. Figure 4.10 was generated using manual pixel reordering.

Manual Selection

Selection tools enable users to select dimensions of interest for further exploration using other multi-dimensional visualization techniques. They can also be used as a filter to reduce the number of subwindows displayed in a pixel oriented display. **Manual selection** allows a user to manually select a dimension by left clicking on its corresponding subwindow. The user can unselect a dimension by clicking on any other subwindow.

4.5 Scaling to Datasets with Large Numbers of Data Items

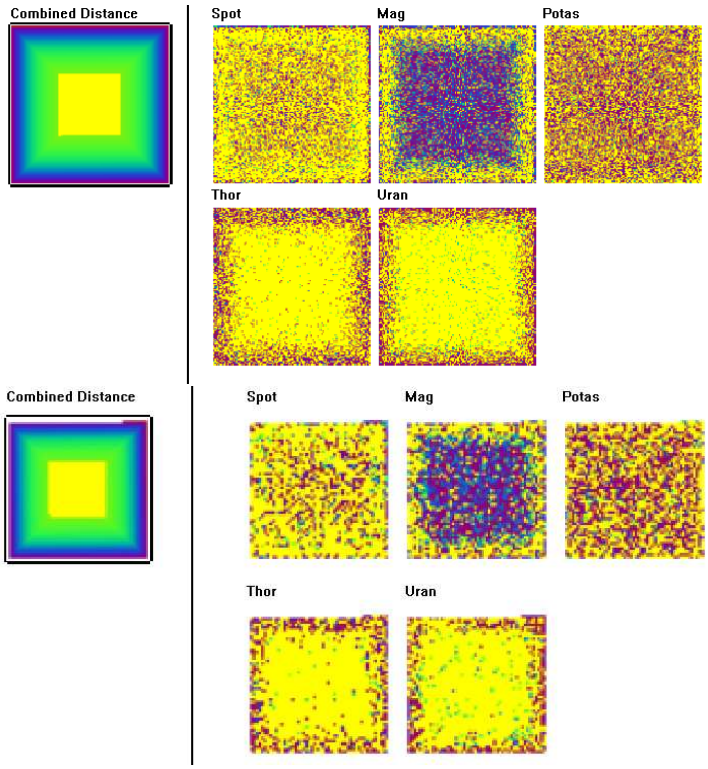

Figure 4.11: A comparison of the display of (a) full dataset and (b) sampled dataset of the remote sensing dataset. Both visualizations seem similar.

When generating a pixel oriented display for a dataset containing a large number of data items, we use a random sampling approach to reduce response time of user interactions. In particular, users can set the sampling factor (between 0 and 1.0) to return a sampled dataset. Figure 4.11 shows two pixel oriented displays of a dataset with sampling (factor = 0.25) and without sampling. It can be seen that the corresponding subwindows in the two displays have very similar patterns.

Secondly, in order to use the capabilities of the graphics card on the user's computer and improve the speed of user interactions for large-scale datasets, we store all subwindows as texture objects in OpenGL. Thus unless we need to regenerate

the texture of the subwindows, each subwindow can be refreshed, repositioned, or resized on the screen by simply redrawing the texture objects, mapping the texture objects to different positions on the screen, or mapping them to areas of different sizes. All these operations can be efficiently performed in hardware.

Both the above two approaches cause information loss in the pixel oriented display. When random sampling is performed, data items not in the sample are not visually presented to the user. When the texture objects are mapped to screen areas that are not exactly their original sizes, magnification or minification happens so that the pixels visualized are only approximations of the original pixels. However, information loss is exchanged for the reduction of clutter in the display and the reduction of response time of user interactions, which are very important for a visual exploration task. Moreover, approximation is usually acceptable in a visualization system. Furthermore, users can always get the information accurately by setting the sampling factor near 1.0, and setting the size of the subwindows to exactly the size of the texture objects.

Chapter 5

Hierarchical Pixel Oriented Visualization

The hierarchical visualization in XmdvTool also addresses the visualization of large data sets by presenting a multi-resolution view of the data. Creation of the hierarchy preserves all the information present in the data while having access to the raw data elements. All the displays in XmdvTool have a counterpart in the hierarchical world and display the aggregate information maintained at each node in the cluster tree. The set of associated hierarchy navigation tools allows the user to discover patterns in the data set and are the topic of discussion for this chapter.

5.1 Hierarchical Clustering

The primary purpose for building a cluster hierarchy is to structure and present the data at different levels of abstraction. A hierarchical approach is a convenient mechanism for organizing large datasets. By recursively clustering or partitioning data into related groups and identifying suitable representative information (sum-

marizations) for each cluster, we can examine the data set methodically at different levels of abstraction, moving down the hierarchy (drill-down) when interesting features appear in the summarizations and up the hierarchy (roll-up) after sufficient information has been gleaned from a particular subtree.

A clustering algorithm groups objects or data items based on measures of proximity between pairs of objects [JD88]. In particular, a hierarchical clustering algorithm constructs a tree of nested clusters based on proximity information.

Let $\mathbf{E}$ be the a set of k N-dimensional objects, i.e.,

$$\mathbf{E} = \{e_1, e_2, e_3, ..., e_k\}$$

where e_i is an N-vector:

$$e_i = \{x_{i1}, x_{i2}, x_{i3}, ..., x_{iN}\}$$

An m-partition $\mathbf{P}$ of $\mathbf{E}$ breaks $\mathbf{E}$ into m subsets $\{P_1, P_2, ..., P_m\}$ satisfying the following two criteria:

$$\begin{aligned} P_i \cap P_j &= \emptyset \quad \text{for all } 1 \leq i, j, \leq m,\ i \neq j\ , and \\ \bigcup_{i=1}^{m} P_i &= \mathbf{E} \end{aligned}$$

A partition $\mathbf{Q}$ is nested into a partition $\mathbf{P}$ if every component of $\mathbf{Q}$ is a proper subset of a component of $\mathbf{P}$. That is, $\mathbf{P}$ is formed by merging components of $\mathbf{Q}$. A hierarchical clustering is a sequence of partitions in which each partition is nested into the next partition in the sequence.

A hierarchical clustering may be organized as a tree structure: Let P_i be a

component of $\mathbf{P}$, and $\mathbf{Q}$ be the m partitions of P_i. Let P_i be instantiated by a tree node T_i. Then, the components of $\mathbf{Q}$ form the children nodes of T_i.

The approaches that impose a hierarchical structure to a data set can be broadly categorized as either *explicit* or *implicit* clustering. In *explicit clustering*, hierarchical levels correspond to dimensions and the branches correspond to distinct values or ranges of values for the dimension. Hence different orders of the dimensions give different hierarchical views. On the other hand, *implicit clustering* tries to group similar objects based on a certain metric, for instance the Euclidean distance.

There is a large body of literature on algorithms for the computation of implicit clusters [JD88]. The particular method used for the tree construction is however not relevant to this thesis. Any method that builds a tree that abides by the above definitions could in principle be used as the tree construction scheme in our system.

However, most clustering algorithms are not appropriate for large data sets because of large storage and computation requirements. In recent years, a number of algorithms for clustering large data sets have been proposed [ADO90, GRS98, ZRL96]. XmdvTool adopts the K-means clustering algorithm [Mac67] although all the visualization techniques would work equally well with data clustered by other methods.

5.2 Query Specification on Hierarchies

We had seen in chapter 4 that the brush directly corresponded to an N-dimensional range query and that the brush could be specified in data-space or screen-space. Hierarchical clustering of the data set introduces another space, termed as structure-space, where brushing could be performed. This section introduces structure-based brushing concepts and then uses these concepts to define an N-dimensional brush from the parameters of the structure-based brush.

5.2.1 Structure-Based Brushing

As noted, a *hierarchy* or a *tree* is a convenient mechanism for organizing large data sets. By recursively clustering or partitioning data into related groups and identifying suitable representative information (summarizations) for each cluster, we can examine the data set methodically at different levels of abstraction, moving down the hierarchy (*drill-down*) when interesting features appear in the summarizations and up the hierarchy (*roll-up*) after sufficient information has been gleaned from a particular subtree.

XmdvTool has a suite of techniques, termed *structure-based brushing*, aimed at supporting the interactive exploration of large data sets that are either implicitly or explicitly clustered into a hierarchical structure.

The first containment criteria for the brush is an augmented monotonic value relative to its parent for each node in the hierarchy, that is each cluster. This value can be, for example, the level number, the cluster size/population, or the volume of the cluster's extents (defined by the minimum and maximum values of the nodes in the cluster). This assigned value determines the control for the level-of-detail. By choosing a continuous control variable, such as cluster size, the traversal of the tree through different levels-of-detail can be smooth transitions instead of abrupt screen changes. Formally, the level-of-detail variable L with range l_{min} to l_{max} and a value range $[l_a, l_b]$ can be defined such that $l_{min} \leq l_a \leq l_b \leq l_{max}$.

The second containment criteria for structure-based brushing is based on the fact that each node in a tree has extents, denoted by the left- and right-most leaf nodes originating from the node. In addition, it is always possible to draw a vertically oriented tree in such a way that the horizontal position of each node (and, in fact, all of its children) falls between its extents. These extents ensure that a selected subspace is contiguous in structure space. Formally, we define an extents variable E,

with range e_{min} to e_{max} and an extent range $[e_c, e_d]$ such that $e_{min} \leq e_c \leq e_d \leq e_{max}$.

Thus a structure-based brush is defined by a subrange of the structure extents and level-of-detail variables, namely $[l_a, l_b, e_c, e_d]$.

5.2.2 Creation and Manipulation of Structure-Based Brush

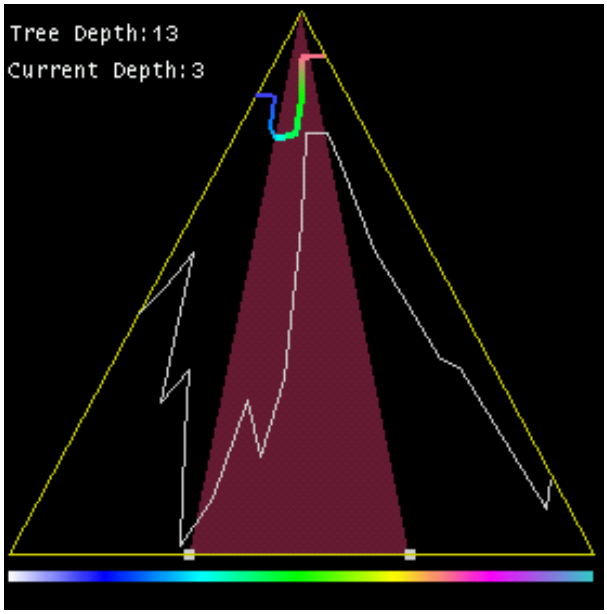

Figure 5.1: Structure-based brushing tool. (a) Hierarchical tree frame; (b) Contour corresponding to current level-of-detail; (c) Leaf contour approximates shape of hierarchical tree; (d) Structure-based brush; (e) Interactive brush handles; (f) Colormap legend for level-of-detail contour.

Figure 5.1 shows the *structure-based brushing interface* in XmdvTool. The triangular frame depicts the hierarchical tree (see (a)). The leaf contour (see (c)) depicts the silhouette of the hierarchical tree. It delineates the approximate shape formed by chaining the leaf nodes. The colored bold contour (see (b)) across the tree delineates the tree cut that represents the cluster partition corresponding to the specified level-of-detail. A *proximity-based coloring* scheme is used in assigning colors to the partition nodes [FWR99a].

In this scheme, a linear order is imposed on the data clusters gathered for display at a given level-of-detail. This linear order is directly derived from the order in which the tree is traversed when gathering the relevant nodes for a given level-of-detail. In our implementation, we adopt the in-order tree traversal. Colors are then assigned to each cluster by looking up a linear colormap table (see (f)). The same colors are used for the display of the nodes in the corresponding data display. The two movable handles (see (e)) on the base of the triangle, together with the apex of the triangle, form a wedge in the hierarchical space (see (d)).

The structure-based brushing tool supports both direct and indirect manipulation. Sets of elements may be directly selected by positioning the wedge handles so as to bound the range of colors spanned by the elements. This is made possible by the direct color correspondence between the data display and the structure display. Moreover, similar elements are selected as a group, since by our coloring criteria, similar elements are drawn in similar colors. The wedge handles can be adjusted at either end or simply translated to bound the desired set of elements. Indirect manipulation is provided through the use of sliders for the range of extents and values, in case the user prefers such a mode of interaction.

In a hierarchical organization, drill-down and roll-up operations are commonly used to explore the hierarchy. Our tool supports a global drill-down and roll-up operation, that is, the current level-of-detail can be adjusted by dragging the colored contour vertically. The data display changes to reflect more details when the contour is adjusted vertically downwards while showing more and more abstract views of the data when the contour is adjusted vertically upwards.

Besides a global drill-down/roll-up operation, our tool also allows the user independent control of the level-of-detail of the brushed and unbrushed region. That is, the colored contour in the brushed region can be adjusted independently of the

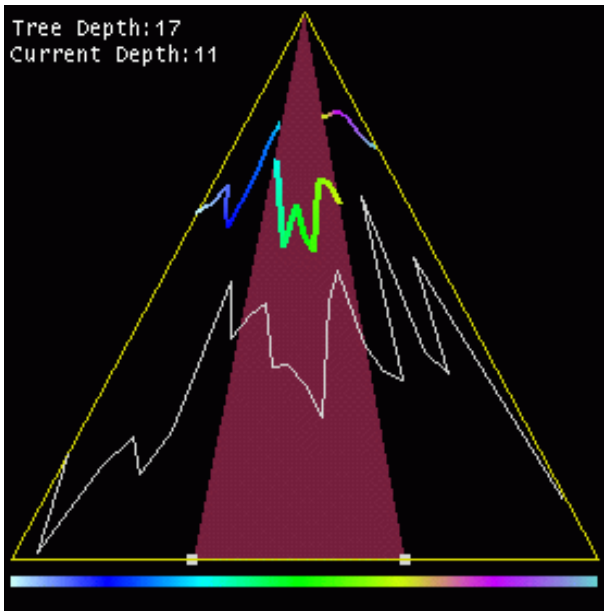

Figure 5.2: Structure-based brushing at two different levels-of-detail.

contour segments outside the brushed region, and vice versa (See example in Figure 5.1). We term this *selective drill-down/roll-up*. This separate mode of control gives the user the flexibility to view the hierarchical structure at two different levels-of-detail at the same time.

5.2.3 Structure Based Ordering of Pixels

In order for hierarchical pixel oriented displays to show the same semantics as the flat pixel oriented displays, it is important to contrast structure-based brushing with traditional data-based brushing. In a traditional user-driven brushing operation, to specify a region of interest in a multivariate data display, the user sets upper and lower bounds for each dimension. In data-driven brushing, the user paints over groupings of interesting data. Neither of these approaches is suitable for isolating data elements which are *structurally* related. Rather, their focus is on the *values* of the data. Clearly, structure-based brushing provides new, and potentially invaluable,

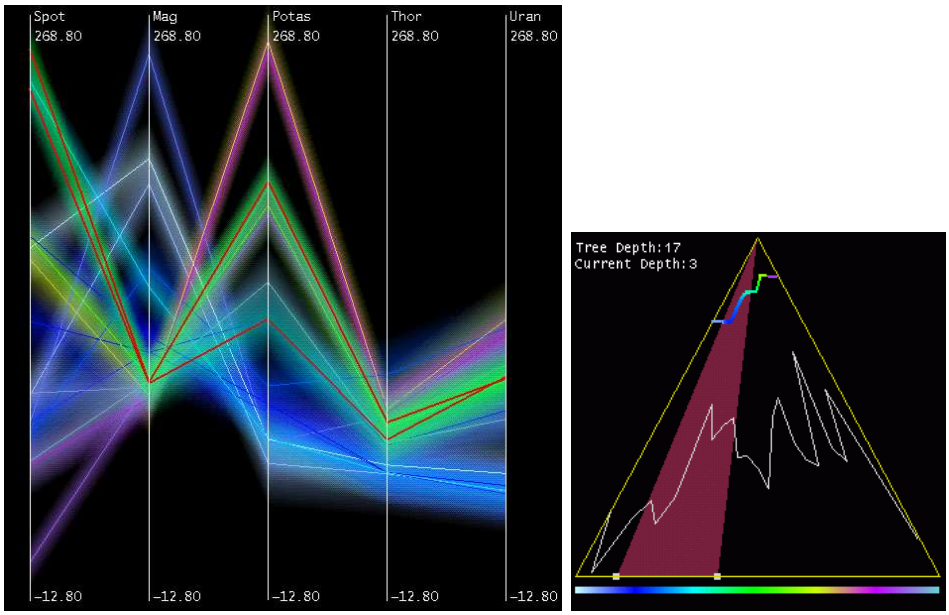

Figure 5.3: A hierarchical parallel coordinates display of a remote sensing dataset with the selected cluster painted in bold red to reflect that it is currently being brushed in the structure-based tool. The image on the right shows the corresponding level-of-detail indicated by the colored contour in the structure-based brush with the brushed region indicated by the wedge. In this case, we observe that the selected clusters share the same mean value for magnetics and uranium contents, and have high SPOT contents.

functionality beyond data-based brushing.

A logical question is whether it is possible and, if so, useful to switch between structure-based and data-driven brushes. Each is a specialization of what we term *set-based brushing*, where every point is classified as either being within the brush or outside the brush. In explicit hierarchies, which are often generated via categorical data or ranges of data values, there is generally a direct mapping from structure-based brushes to data driven brushes, but not vice versa. A user could specify a subspace of the hierarchy that maps to a contiguous subspace in data space. This would correspond directly to a data-driven brush. A simple method to implement this would be to collect all clusters selected by the specified extents of the structure based brush and define the smallest N-dimensional region containing all the selected

clusters.

We use this direct mapping from structure-based brush to a traditional N-dimensional data brush to generate the Hierarchical Pixel Oriented Display. The structure-based brush drives the selection of clusters at the specified level of detail. Distances are calculated for the all the clusters for the current level of detail. The overall distance can be calculated as in the case of flat pixel oriented displays, which can order the pixels of the display. We term this ordering controlled by the structure based brush as Structure-based Ordering.

5.3 Visualizing Clusters

The hierarchical pixel oriented display is the new extension that we have developed for visualizing large data sets using pixel oriented techniques. In the hierarchical pixel oriented display, data is structured as a hierarchy of clusters, and the display shows summarizations of the clusters at a certain level of detail. Many display options exist, including showing cluster centers (which look identical to traditional flat pixel oriented displays), cluster population (which manifest themselves as replicated pixels in each subwindow) and other cluster statistics. Distortion techniques as in the flat display, structure-based coloring, and selective fade-in/fade-out are available to help reduce clutter and expose structure.

Figure 5.4 shows the hierarchical pixel oriented display at a given level-of-detail. Each pixel across the subwindows displays the mean value of its cluster. The size of the subwindows gives an indication to the number of clusters at the given level-of-detail. For the cluster population display option, the color of the pixel is replicated by the number of leaves in the corresponding cluster. The coloring of the pixels is according to the techniques listed for flat pixel oriented displays. With the brushing

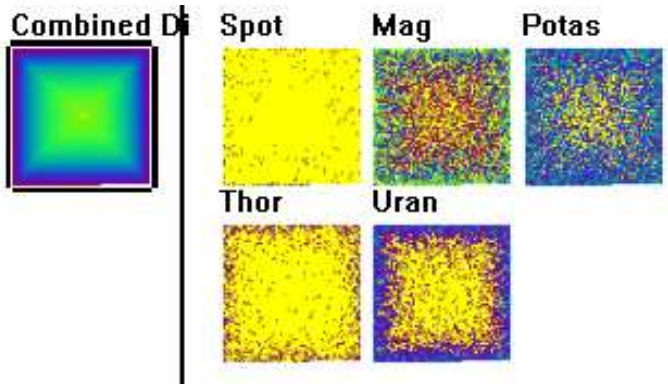

Figure 5.4: Hierarchical Pixel Oriented Display.

tool, the user simply adjusts the handles at the base of the triangle wedge to bound the extents of interest. The selected clusters are drawn in yellow, indicating they are being brushed. Next, we demonstrate the usefulness of the selective drill-down/roll-up operations.

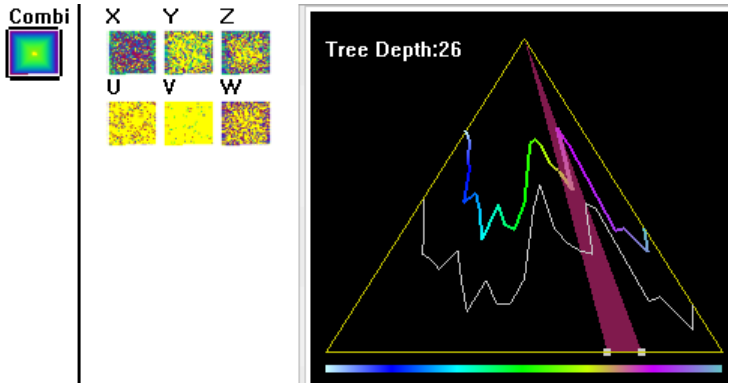

Figure 5.5: Hierarchical Pixel Oriented Display at lod level of 0.2 for both brushed and unbrushed clusters for the UVW (6 dim - 150,000 data points)

Figures 5.5 and 5.6 show two images of hierarchical pixel oriented displays at different levels-of-detail for the UVW dataset which is a 6-dimensional dataset containing around 150,000 data points. Figure 5.5 displays the initial state, with all clusters at the same level-of-detail. The user can then brush the cluster(s) of interest by adjusting the handles at the bottom of the wedge on the structure-based brush interface. Next, by "pulling" the brushed contour vertically downwards, we can view

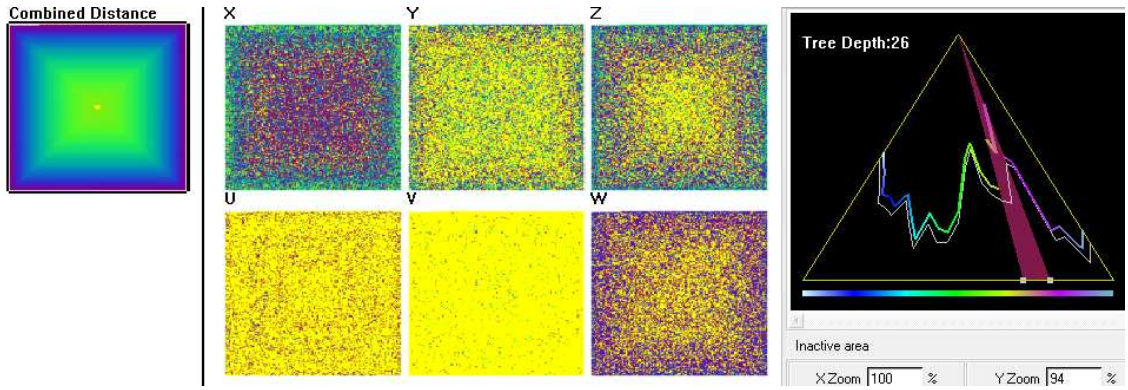

Figure 5.6: Hierarchical Pixel Oriented Display with brushed clusters at lod level of 0.2 while unbrushed clusters at LOD level of 0.06. The dataset is the UVW Dataset

the unselected clusters at higher level-of-detail while maintaining the same level-of-detail for the selected clusters. This results in the display shown in Figure 5.6, which helps the user in identifying how the unselected region relates to the query. The usefulness of the selective drill-down/roll-up feature is evident here – users have the flexibility to see relationships between the clusters at the region of interest. One important observation is that the textures in Figures 5.5 and 5.6 almost look the same. This indicates that the relationships are maintained across resolutions (which is to be expected). Thus, the user can opt to use a lower resolution for exploring the dataset and then switch to higher resolution for higher accuracy.

The subwindows of the hierarchical pixel oriented display contract when rolling up the level of detail brush because the size of the subwindows directly correspond to the number of clusters being visualized. We have implemented the population display option for hierarchical pixel oriented display which replicates the pixels for a particular cluster by the number of leaves in the cluster. This maintains the size of the subwindows, but loses correspondence with the level of detail. Figures 5.7 and 5.8 show the extents visualization for the UVW Dataset.

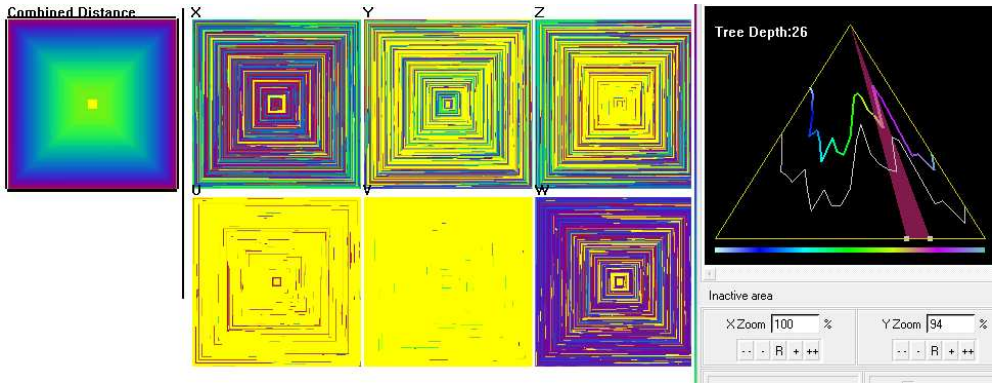

Figure 5.7: Hierarchical Pixel Oriented Display (Extents Visualization) with brushed clusters at LOD level of 0.2 while unbrushed clusters at LOD level of 0.06. The dataset is the UVW Dataset

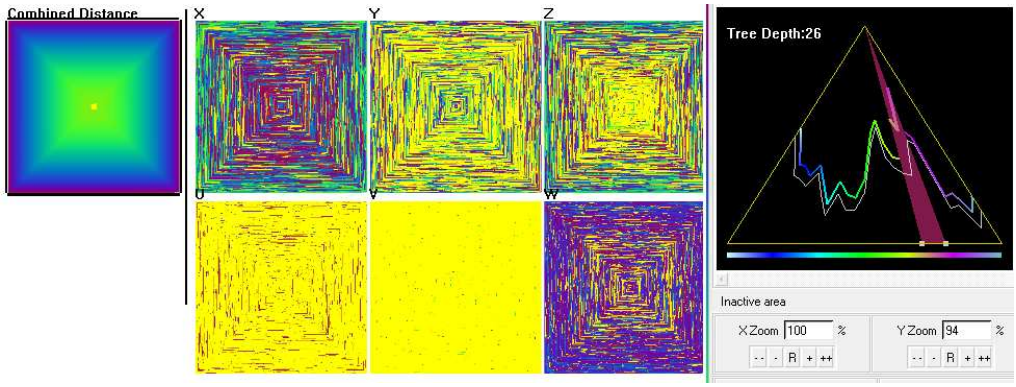

Figure 5.8: Hierarchical Pixel Oriented Display (Extents Visualization) at LOD level of 0.2 for both brushed and unbrushed clusters for the UVW (6 dimensional - 150,000 data points)

Chapter 6

Implementation

This chapter briefly describes the design and current implementation of the flat and hierarchical pixel oriented displays with respect to the XmdvTool system.

6.1 Platform

The new display modules were implemented as part of the XmdvTool system version 6.0a. The system is a cross-platform visualization toolkit and works on Windows 2000/XP and Linux platforms. The language of implementation is C++, which is a superset of the C programming language that includes object-oriented features. The software development process of this project is completely object oriented, aided by the fact that the XmdvTool system is itself organized using object-oriented methodologies. C++ is also the language of choice for high-performance applications. The application interacts with the underlying graphics platform through Tcl/Tk [tcl] for GUI and OpenGL [ope] for rendering. This makes it completely platform independent, flexible to add GUI widgets and takes advantage of the graphics hardware capabilities of the underlying machine.

6.2 Design

This project started as a prototype for incorporating pixel oriented visualizations in XmdvTool which was developed as part of the Data Visualization course (CS525D) in Spring 2003. As the project grew into a thesis implementation, the goal was to provide a framework so that the development of different types of pixel oriented techniques could be integrated into the system.

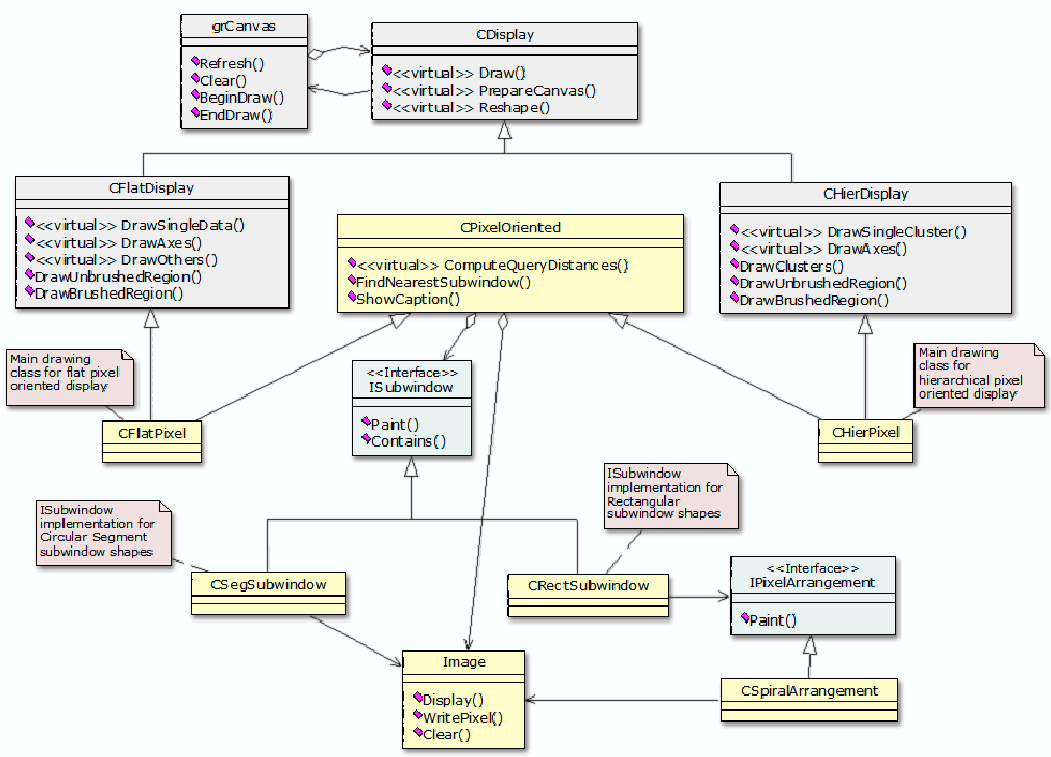

Figure 6.1: Class Diagram

Figure 6.1 shows the class diagram for the classes that were integrated into the system. The system provides a basic framework of classes (*grCanvas*, *CDisplay*, *CFlatDisplay*, and *CHierDisplay*) for adding new display types to the tool. *grCanvas* handles the OpenGL interface for all the displays. The main application maintains a list of *CDisplay* objects that would correspond to the display modules in XmdvTool. The pixel oriented display module consists of the main base class

CPixelOriented that provides functions common to the two types of pixel oriented displays (*CFlatPixel* and *CHierPixel*). *CPixelOriented* also maintains an $N+1$ number of *ISubwindow* objects for an N-dimensional dataset to visualize the distance values of each dimension and the combined distance. The *ISubwindow* interface allows implementation of different shapes of the subwindows. Two implementations of this interface have been provided – *CRectSubwindow* for rectangular subwindows and *CSegSubwindow* for circular segments. The *CRectSubwindow* also allows for different types of pixel arrangements through the *IPixelArrangement* interface, an example being the *CSpiralArrangement* class. The Strategy design pattern has been used for the shape and pixel arrangement classes to allow for more techniques to be added in the future.

Dialogs that were added or modified for the new pixel oriented modules were:

- Pixel Oriented Brush Toolbox dialog was added, which is similar in functionality to the other brush toolboxes for the respective displays in XmdvTool.
- Weight Modification dialog that shows sliders for assigning weights for each of the dimensions.
- ColorMap Editor for changing the standard brushed and unbrushed colormaps. The ColorMap Editor can be accessed via the existing Color Requester dialog which is already being used for changing system-wide colors in XmdvTool.

6.3 Issues

During the development of the pixel oriented framework, many issues were encountered. Some of the issues related to the flat pixel oriented display were:

- Query specification was one of the main issues to be resolved. The main constraints were that the controls should not take up a lot of screen space. We provide a limited query specifier through a colormap display for the currently selected dimension on the pixel oriented display and an extensive one through colormaps for all dimensions on an auxiliary dialog.

- Specifying queries through the colormap display can be thought of as screen-space brushing or user-driven brushing (Refer section 4.3.1). Data-space brushing, provided in other displays in XmdvTool, was a tricky issue for pixel oriented displays. Selecting a data point which is just one pixel wide would be very difficult for users. This required distorting the view and providing some control over the selection process. Different distortion techniques were considered, such as providing a zooming window or fish-eye views. Finally, a focus+context type of distortion was identified and applied to the display. For data-space brushing, options such as providing a slider on mouse click over a pixel or using the distance of the mouse after a click and drag were considered. None of these proved to be intuitive to the user. Finally, a paint-like operation, where the user presses shift+mouse click to paint over the pixels to specify the data points contained in the selection, was developed and seemed to work well.

- All the rectangular subwindows were arranged in a raster-like fashion for the non-MDS placement technique. It was felt that placing the derived dimension (the Combined Distance dimension) in line with the other dimensions might confuse the user. Hence, we now display the subwindows for all the derived dimensions (there might be more in the future) separately. A line is drawn to separate the two classes of subwindows and we have also added the capability

to toggle the display of the derived dimensions.

- The earlier prototype displayed the subwindows with a colormap for the selected dimension, which allowed for brush specification. However, it was later realized that the colormap could also be toggled to allow for more screen space for the subwindows. When the colormap is not shown, the brush can be specified through any of the auxiliary brush toolboxes in XmdvTool or via the paint-like operation introduced earlier.

- A GUI was developed using the GLGooey [glg] package to allow setting up the various options and bringing up the dialogs from the display itself. This allowed a centralized control over the pixel oriented display and its related functionalities.

Some of the issues related to the hierarchical pixel oriented display were:

- The structure-based brush already provides an ordering for the clusters. The order assigned by the cluster tree (left to right) is completely dependent on the clustering algorithm. The proximity based coloring provided by the other displays can conveniently use this ordering and use a simple colormap to show the cluster relationships. We felt that although this was useful for the other displays in XmdvTool, this restriction was too tight to take advantage of providing query feedback in pixel oriented techniques. Hence, we have structure-based ordering to induce a query-based ordering to the clusters for the benefit of providing query relevance visualization for hierarchies.

- A variety of cluster statistics are possible for the hierarchical pixel oriented display such as cluster means, extents, population or distribution. The most intuitive option seemed to be to display cluster means (as single pixels) as is

the default option in other displays. The level-of-detail controls the size of the subwindows. The main issue was to port the mean-band technique (used for showing cluster extents in other displays) to hierarchical pixel oriented display, which meant using opacity and pixel replication to display the cluster extents. Adding opacity levels to pixels sprinkled over the subwindow would lead to confusion, and hence, population is displayed instead of extents. The population of the cluster controls the number of pixels that are to be replicated for each cluster.

Chapter 7

Evaluation

We have conducted a small user study to illustrate the usefulness of adding pixel oriented displays in XmdvTool. This chapter briefly describes the goal, the method and the result of this evaluation process.

7.1 Introduction

The goal of pixel-oriented displays is to use every available pixel to show as much data as possible on the screen thereby providing local detail in the global context. This type of representation aims to solve the clutter problem associated with information dense displays. Usability tests for pixel-oriented displays have been conducted [KK96] for their perceptual properties and data usability features. Pixel-oriented visualization techniques have been shown [KK95] to be useful for the exploration and analysis of large databases to find interesting data clusters and their properties such as functional dependencies and correlations among the dimensions.

The power of this representation can be magnified by combining it with various interaction/distortion techniques and linked views. Pixel oriented displays provide an overview of the data while the other displays in XmdvTool provide detailed views

of the data. Pixel oriented displays also provide a query feedback mechanism where it not only plots the results of the query, but also the relevance of the unselected data points with respect to the query. Chimera's study [CWMS94] seems to indicate that overview-and-detail should perform better than detail-only. The goal of this study is to measure the added value of coordinated pixel-oriented and other visualizations in terms of user task times and subjective satisfaction for browsing large information spaces.

The questions to guide us on this quest are:

- How does the multiple linked view with pixel oriented display and the interactions/distortions introduced in this display help the user to specify "good" queries?
- The visual feedback across the visualizations could be distracting or disorienting for the users. But, if there is a benefit, what is its magnitude?
- Which enhancement is the important factor that causes improved user performance? Is it the information displayed in the overview or the coordination between the overview and detail?
- What are the successes and shortcomings of the flat pixel oriented displays integrated into XmdvTool?

7.2 Finding Data Characteristics in Pixel Oriented Displays

Before presenting our evaluations, in the following we briefly introduce how data characteristics such as correlations, functional dependencies, and clusters may be

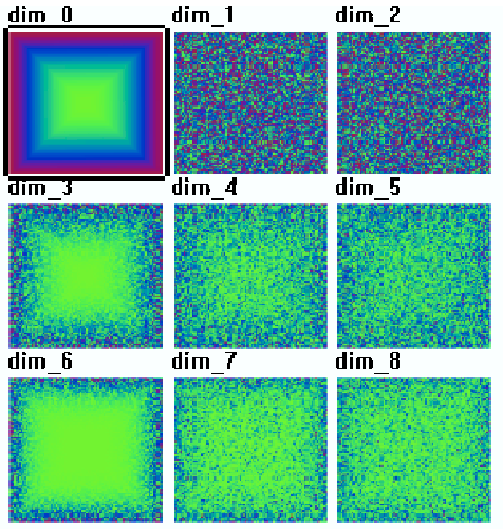

Figure 7.1: Linear functional dependency between dimensions 0 and 3 and quadratic functional dependency between dimensions 0 and 6. Note that there is no dependency between dimension 0, 1 and 2 as they are completely dissimilar in appearance.

identified in our visualizations generated by the spiral technique. Note that these techniques are interactive in nature and are therefore difficult to describe in written form.

Properties that hold for all or most of the data can be deduced from the overall brightness and color distribution of the visualizations. The size of the yellow portion of the visualization indicates the number of data items fulfilling the query for the corresponding attribute. The brightness of the visualization of some attribute indicates the degree of fulfilling the corresponding query predicate, and the overall color distribution shows the distribution of distance values for the corresponding attribute. Sharp borders between colors, which indicate discontinuities in the value range of an attribute, are especially interesting.

Usually the visualizations of the attributes are not independent. Using our visual data mining techniques, correlations and functional dependencies between attributes may be identified by the similarity of their visualizations (see Figure 7.1). In cases

where existing structure in the data is not revealed by the visualization, the structure can usually be made visible by arbitrarily changing the weighting factors of some attributes. Changing the weighting factors makes it easier to perceive similarities in the visualizations and to determine correlations and functional dependencies.

Another important group of interesting data characteristics is clusters of data items with similar properties. In our visualizations, clusters usually appear as rectangular regions (possibly partial rectangles) which may have different colors for different attributes (See Figure 7.2). Clusters may have a lower dimensionality than the whole data set. Therefore, the clustering may appear only in the visualizations of certain attributes. Note that the visualizations generated by the spiral technique in general depend on the chosen query region. Changing the query region has a major impact on the resulting visualizations. For example, if the query region is moved away from a cluster, the cluster becomes less visible in the visualization. By interactively changing the query region, different clusters can be made visible.

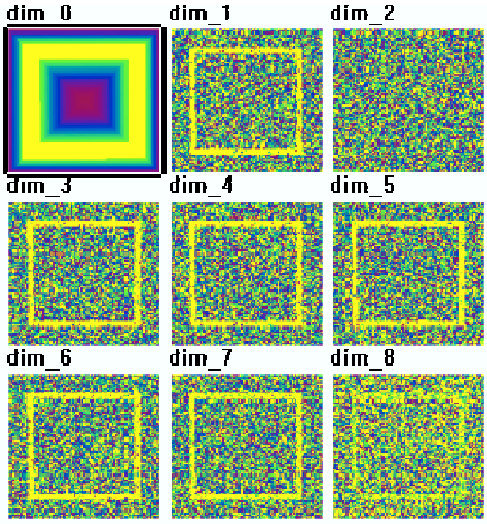

Figure 7.2: Clustering can be easily seen in this view. Also note that the visualization tells us that the cluster is of a lower dimension than the dataset.

7.3 Data

We use a general test data generation model which is used for a standardized and quantitative testing of visualization techniques [BKP94]. The model allows generation of data sets with characteristics similar to those of real data sets. However, the characteristics can be varied arbitrarily. We developed a tool, *DataGen*, that partially implements the test data generation model and allows specification of a wide range of data characteristics. The following five data sets were used for the evaluation:

1. Data set to evaluate finding a 4-dimensional cluster in a 6-dimensional data. The data set consists of 15,000 data items. Two thirds of the data is generated randomly in the range of [0,100] for each dimension and the remaining one third of the data defines 3 clusters that are generated by inserting additional data items in specific range values.

2. We use a data set generated by using different distributions for defining a cluster. The base data set consists of 10,000 data items, uniformly distributed in the range of [-10000,10000] for each of the dimensions. The cluster consists of 1,000 data items and the cluster dimensions differ in the distribution.

3. The third data set is generated using different functional dependencies for defining a cluster. The base data set again consists of a 10,000 data items with 9 dimensions, uniformly distributed in different ranges: [0,1000] for dimensions one to three, [0,2000] for four to six and [0, 1,000,000] for the rest. The functional dependencies of the other dimensions for a cluster of 2000 data items are given in Table 7.1.

4. The fourth dataset is generated with functional dependency as above for the

attribute	1	2	3	4	5	6	7	8	9
Functional dependency	Uniform distribution in the range [0,1000]			Linear dependency of attribute(s)			Quadratic dependency of attribute(s)		
				1	2,3	1,2,3	1	2,3	1,2,3

Table 7.1: Value ranges and dependencies for data items in the cluster of third data set.

complete dataset. The data set is again 10,000 data items with 9 dimensions with the values generated as in the Table 7.1.

5. The last data set used was the Remote Sensing data set, which is a real data set. The users were unaware of which data sets were real and which were artificial. The aim was to see if the users could find any relationships in this data set. The base data set consists of 16,000 data items with 5 dimensions.

Examples of visualization that result from data sets 1 and 4 are provided in Figures 7.3 and Figures 7.4.

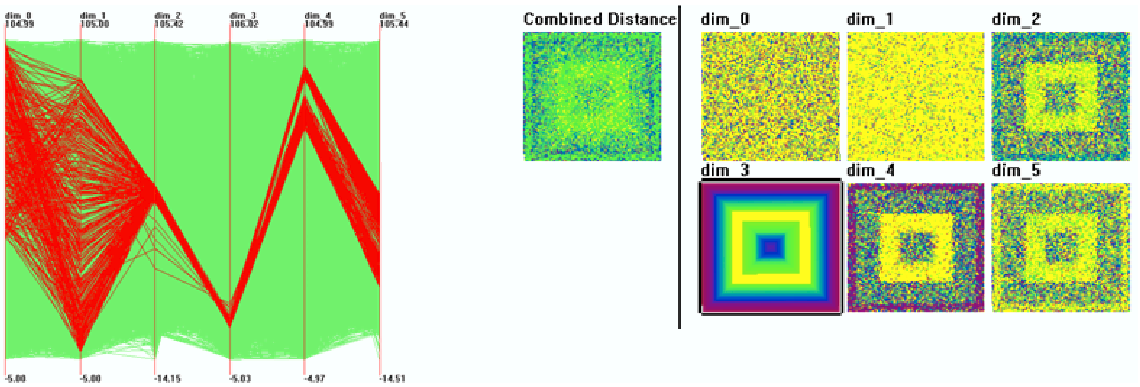

Figure 7.3: Pixel oriented visualization for the first data set and corresponding visualization in Parallel Coordinates. Note that the pixel oriented display shows the 4-D clusters.

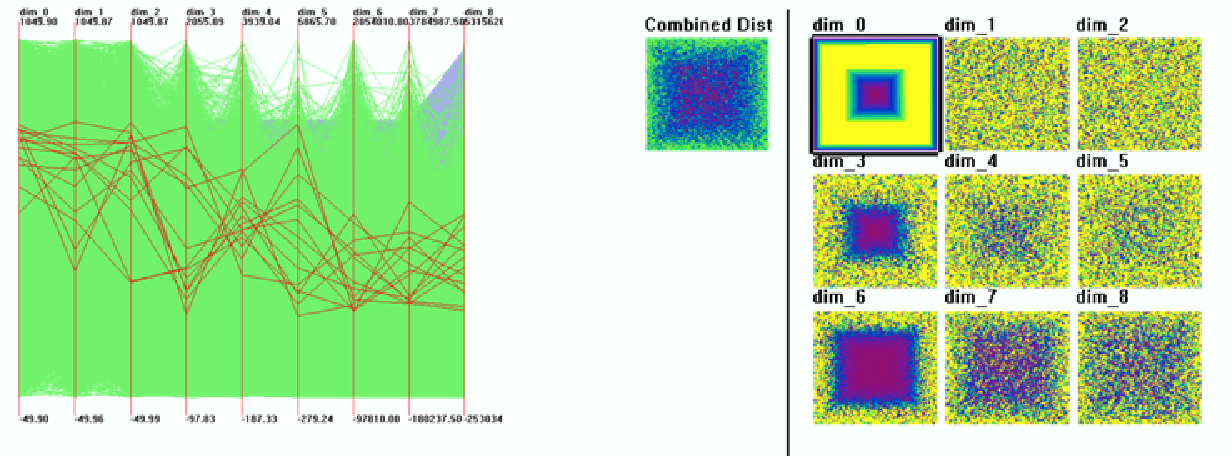

Figure 7.4: Pixel oriented visualization for the fourth data set and corresponding visualization in Parallel Coordinates. Note that the pixel oriented display shows correspondence between dimensions 0, 3 and 6.

7.4 Variables

7.4.1 Independent Variables

A variety of browsing tasks, using a question-answer approach were presented to the subjects. Some of the treatments are as follows:

- How many clusters can you see in the data set? What is the dimensionality of the observed cluster? Was the cluster visible before/after discovered in pixel oriented display?
- Are any correlations seen between the dimensions? What type of correlation is it (Linear/Quadratic)?

The complete list of tasks for each of the data set is provided in Appendix A.

7.4.2 Dependent Variables

As part of the evaluation process, we measured two dependent variables, namely user performance time and user subjective satisfaction.

- **User performance time:** Time to correctly complete each task, not including reading the task question.

- **User subjective satisfaction:** Subjects rate their satisfaction with each interface on a scale of 1 to 10 on four categories (with scales): comprehensibility (confusing to clear), ease of use (very difficult to very easy), speed of use (very slow to very fast), overall satisfaction (terrible to wonderful).

7.5 Procedure

There were 3 users for the evaluation. All the 3 students were Computer Science students who were also familiar with the XmdvTool system. All of them were familiar with the concepts of using multivariate visualizations for the purpose of finding relationships between dimensions and cluster finding. All of them had some idea of pixel oriented visualization but only one of them had worked with this technique before. The concepts of pixel oriented visualization, objective of the test and procedure were explained to each user. Each of them were asked to work with the system for approximately 10 minutes. The users were asked to note the approximate time for the evaluation of each data set and save the generated image for the pattern they had discovered.

7.6 Results and Analysis

Most of the users were able to find the clusters and functional dependencies in the data set. The users found that clustering was easier to detect in pixel oriented visualization than in other displays, but analysis of the results shows that this also depended on the size and dimensionality of the cluster. Users could find linear

dependencies among the dimensions in the data set, but found it difficult to find quadratic dependencies. Also, a small cluster with function dependency was difficult to discover than larger clusters. The users found changing weights of the dimensions to be confusing and was used sparingly. Users could also find some interesting clustering in the Remote Sensing data set through pixel oriented displays.

It was noted that the time taken for completing each of the tasks reduced as they progressed in the evaluation. This may be attributed to increasing familiarity with the system during the course of the evaluation. The rating of all the users were on the high side (7-9) for all the subjective variables. They found the system to be quite responsive as compared to other displays in XmdvTool.

Overall, the experience of the evaluation was good and we received some feedback on improving the system and the evaluation. Some users noted that the provided brushing interaction was intuitive and unique. They suggested some strategies such as adding a simple button to change the query regions to the full or zero range in all dimensions. One user noted that an option for turning off the highlighting and using query-independent technique could be provided to enhance the display.

7.7 Conclusion

The above results does seem to indicate that pixel oriented visualization does increase the productivity of the user when used along with other displays in XmdvTool. The study concludes that the ability to visualize large data sets in maximum resolution and find patterns in the data is enhanced when pixel oriented visualization is used in conjunction with other multivariate techniques.

Chapter 8

Conclusions and Future Work

Visualization is becoming a necessity in today's academic, government and commercial working world. Data is becoming more and more complex, and increasing in amount. The ability to see and understand the data to make critical decisions is needed now more than ever. The need to share this data in a meaningful way, where one can interact with others and draw important conclusions, is technology one has to embrace. Many multivariate visualization tools exist that allow one to analyze the data, but most of them fail to display large data sizes which manifest as clutter on the display. One way to visualize these large data sizes is using the optimum display resolution today's graphics hardware and monitors can offer and pixel oriented techniques aim to accomplish that.

8.1 Realization of Goals

In this project, we proposed to implement and link pixel oriented visualization techniques with the other display techniques in XmdvTool. Pixel oriented visualization techniques are not new and there are already implementations available. However, to the best of our knowledge, this project is the first attempt at linking pixel ori-

ented visualizations with other multivariate displays and introducing hierarchical pixel oriented displays for larger data sizes. These techniques also offer a better query feedback than any other multivariate techniques, so that the user is better equipped to explore and mine the data.

In this project, we have implemented an enhanced version of the pixel oriented display by working out the issues with existing implementations. The efforts have largely been concentrated on providing a framework for pixel oriented displays in XmdvTool so that it can be easily extended. This includes providing a suite of navigation and interaction tools, such as zooming, panning, distortion, and brushing, for the display. Many new issues have been discovered and implemented during the course of development of pixel oriented displays. Such issues include placement of the subwindows in a way that provides more information about the relationships of the dimensions, and sampling of the data set to improve the handling of large data sizes in the flat display itself. The newer pixel oriented display in XmdvTool also borrows ideas such as screen-space and data-space brushing from the other displays in XmdvTool. This common interaction provided in all XmdvTool displays allows seamless linking between the different views, which was one of the main ideas in including pixel oriented techniques in XmdvTool.

This project also introduced a completely new pixel oriented display for hierarchical data structures. The hierarchical pixel oriented display is the first of its kind in XmdvTool that provides query feedback for the structure-based brush. Using this kind of multi-resolution technique for pixel oriented displays allows one to load very large data sets (sizes in the order of tens to hundreds of thousands) and visualize cluster relationships in the hierarchy.

8.2 Future Work

Directions for future research include both enhancing the current approach and making it more efficient. In the future, we plan to pursue the following tasks:

- Functionally extend the system by integrating more forms of pixel arrangement techniques. A number of generalized spiral and axes techniques have been suggested by Keim et al [Kei00] that improve upon the spiral technique implemented in XmdvTool. The current implementation is flexible enough to allow different forms of pixel arrangement techniques to be integrated within the framework.

- Allowing application of multiple brushes or query regions to pixel oriented displays. This might involve showing multiple layers of subwindows for each of the query regions. A query independent display might be required for displays with no brushes.

- Improve the hierarchical pixel oriented display by including more statistics about the cluster on the display such as cluster extents or cluster distribution. A separate display method is required for showing the level-of-detail (as a separate subwindow or through separate color mapping or maybe through a separate 3D display of the subwindows).

- Improve the efficiency of the system by carefully studying and analyzing the bottlenecks, such as computation of the distance to the query region, in the system and optimizing them. This is necessary for pixel oriented visualizations to be viable for very large datasets (from millions to tens or hundreds of millions).

- Evaluating the flat and hierarchical pixel oriented displays through more thorough user studies and experiments.

Appendix A

Tasks for Evaluation

1. Are you able to see any clustering in the pixel oriented display (you can use the interactions provided on the pixel oriented display for this)?

2. If yes for question number 1, then save the image and answer the following questions -

   (a) How many clusters can you see?

   (b) What dimensional cluster for each of the clusters can you see?

   (c) Is this information available from any of the other flat displays without using the pixel oriented display? If yes, which one?

   (d) After using the flat pixel oriented, can you see clustering in any of the other displays? If yes, which one?

   (e) Can you see any correlations / functional dependencies among the dimensions for the discovered clusters (you can try changing weights of the dimensions for this)?

   (f) If yes for question number 2e, then save the image and answer what kind of dependency is it (Linear or Quadratic) and what are the weights chosen

to find it?

3. Is there any relationship (correlation / functional dependencies) that you can discover among the dimensions?

   (a) If yes for question number 3, then save the image and answer what kind of dependency is it (Linear or Quadratic) and what are the weights chosen to find it?

4. Approximate Time taken to complete the above tasks.

5. How would you rate this task:

   (a) Comprehensibility (1 - confusing to 10 - clear)

   (b) Ease of use (1 - very difficult to 10 - very easy)

   (c) Speed of use (1 - very slow to 10 - very fast)

   (d) Overall satisfaction (1 - terrible to 10 - wonderful)

Bibliography

[ABK98] M. Ankerst, S. Berchtold, and D. A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proc. of IEEE Symposium on Information Visualization, InfoVis'98, p. 52-60*, 1998.

[ADO90] P. Andreae, B. Dawkins, and P. O'Connor. Dysect: An incremental clustering algorithm. *Document included with public-domain version of the software, retrieved from Statlib at CMU*, 1990.

[AKK96] M. Ankerst, D. A. Keim, and H.-P. Kriegel. Circle segments: A technique for visually exploring large multidimensional dataset. *Proc. Visualization '96*, 1996.

[And72] D. F. Andrews. Plots of high-dimensional data. *Biometrics*, 29:125–136, 1972.

[BC87] A. Becker and S. Cleveland. Brushing scatterplots. *Technometrics, Vol 29(2), p. 127-142, 1987*, 1987.

[Bed90] J. Beddow. Shape coding of multidimensional data on a microcomputer display. *Proc. Visualization '90*, pages 238–246, 1990.

[BKP94] R. D. Bergeron, Daniel A. Keim, and R. Pickett. Test data sets for evaluation data visualization techniques. *Perceptual Issues in Visualization*, page 922, 1994.

[BSB82] J. Bertin, P. Scott, and W.J. Berg. *Graphics and Graphic Information-Processing*. Walter de Gruyter, Inc., 1982.

[Che73] H. Chernoff. The use of faces to represent points in k-dimesional space graphically. *Journal of the American Statistical Association*, 68:361–368, 1973.

[CWMS94] R. Chimera, K. Wolman, S. Mark, and B. Shneiderman. An exploratory evaluation of three interfaces for browsing large hierarchical tables of contents. *ACM Transactions on Information Systems, 12, 4*, pages 383–406, 1994.

[FB90] S. Feiner and C. Beshers. Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. *Proceedings ACM Symposium on User Interface Software and Technology*, pages 76–83, 1990.

[Fur86] G.W. Furnas. Generalized fisheye views. *Proc. of Computer-Human Interaction '86, p. 16-23*, 1986.

[FWR99a] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. *Proc. of Visualization '99, p. 43-50*, Oct. 1999.

[FWR99b] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Navigating hierarchies with structure-based brushes. *Proc. of Information Visualization '99, p. 58-64*, Oct. 1999.

[FWR00] Y. Fua, M.O. Ward, and E.A. Rundensteiner. Structure-based brushes: A mechanism for navigating hierarchically organized data and information spaces. *IEEE Visualization and Computer Graphics, Vol. 6, No. 2, p. 150-159*, 2000.

[glg] GLGooey - OpenGL GUI Library. http://glgooey.sourceforge.net/.

[GRS98] S. Guha, R. Rastogi, and K. Shim. Cure: an efficient clustering algorithm for large databases. *SIGMOD Record, vol.27(2), p. 73-84*, June 1998.

[HL92] G. T. Herman and H. Levkowitz. Color scales for image data. *Computer Graphics and Applications*, pages 72–80, 1992.

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. *Proc. of Visualization '90, p. 361-78*, 1990.

[JD88] K. Jain and C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[JKL77] A. K. Joshi, S. J. Kaplan, and R. M. Lee. Approximate responses from a data base query system: Applications of interferencing in natural language. *Proceedings of the 5th Internation Joint Conference on Artificial Intelligence (IJCAI)*, pages 211–212, 1977.

[Kap82] S. J. Kaplan. Cooperative responses from a portable natural language query system. *Artificial Intelligence*, 19:165–187, 1982.

[KD94] D.A. Keim and H.P. Driegel. VisDB: Database exploration using multidimensional visualization. In *Computer Graphics & Applications*, pages 40–49, Sept. 1994.

[Kei96] D. A. Keim. Pixel-oriented visualization techniques for exploring very large databases. *J. Computational and Graphical Statistics*, 5(1):58–77, 1996.

[Kei00] D. A. Keim. Designing pixel-oriented visualization techniques: Theory and applications. In *IEEE Transactions on Visualization and Computer Graphics*, volume 6, pages 1–20, January-March 2000.

[KK95] D. A. Keim and H.-P. Kriegel. Issues in visualizing large databases. *Proc. Conf. on Visual Database Systems (VDB-3)*, pages 203–214, March 1995.

[KK96] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: A comparison. *Transactions on Knowledge and Data Engineering, Special Issue on Data Mining*, 8(6):923–938, 1996.

[KKA95] D. A. Keim, H.-P. Kriegel, and Mihael Ankerst. Recursive pattern: A technique for visualizing very large amounts of data. In *Proc. of the 6th IEEE Visualization Conference*, pages 279–286, 1995.

[KL92] D. A. Keim and V. Lum. Visual query specification in a multimedia database system. *Proc. Visualization '92*, pages 194–201, 1992.

[KW78] J.B. Kruskal and M. Wish. *Multidimensional Scaling*. Sage Publications, 1978.

[LA94] Y.K. Leung and M.D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Transactions on Computer-Human Interaction Vol. 1(2), June 1994, p. 126-160*, 1994.

[Lev91] H. Levkowitz. Color icons: Merging color and texture perception for integrated visualization of multiple parameters. *Proc. Visualization '91*, 1991.

[LWW90] J. LeBlanc, M.O. Ward, and N. Wittels. Exploring n-dimensional databases. *Proc. of Visualization '90, p. 230-7*, 1990.

[Mac67] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability*, 1:281–297, 1967.

[Mot90] A. Motro. FLEX: A tolerant and cooperative user interface to databases. *IEEE Transactions on Knowledge and Data Engineering*, 2(2):231–246, 1990.

[MRC91] J. Mackinlay, G. Robertson, and S. Card. The perspective wall: detail and context smoothly integrated. In *Proc. ACM CHI, p. 173-179*, 1991.

[MW95] A.R. Martin and M.O. Ward. High dimensional brushing for interactive exploration of multivariate data. *Proc. of Visualization '95, p. 271-8*, 1995.

[MZ92] F. Marchak and D. Zulager. The effectiveness of dynamic graphics in revealing structure in multivariate data. *Behavior, Research Methods, Instruments and Computers*, 24(2):253–257, 1992.

[ope] OpenGL Graphics Rendering Library. http://www.opengl.org/.

[PG88] R. M. Pickett and G. G. Grinstein. Iconographic displays for visualizing multidimensional data. *Proc. IEEE Conf. Systems, Man and Cybernetics*, pages 514–519, 1988.

[RAEM94] W. Ribarsky, E. Ayers, J. Eble, and S. Mukherjea. Glyphmaker: Creating customized visualization of complex data. *IEEE Computer, Vol. 27(7), p. 57-64*, 1994.

[RMC91] G. Robertson, J. Mackinlay, and S. Card. Cone trees: Animated 3d visualization of hierarchical information. *Proc. of Computer-Human Interaction '91, p. 189-194*, 1991.

[SA82] R. Spence and M. Apperly. Database navigation: An office environment for the professional. *Behavior and Information Technology*, 1(1):43–54, 1982.

[Shn92] B. Shneiderman. Tree visualization with tree-maps: A 2d space-filling approach. *ACM Transactions on Graphics, Vol. 11(1), p. 92-99*, Jan. 1992.

[Shn94] B. Shneiderman. Dynamic queries for visual information seeking. Technical Report UMCP-CSD CS-TR-3022, College Park, Maryland 20742, U.S.A., 1994.

[SM83] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, New York, 1983.

[tcl] Tcl/Tk Scripting Language. http://www.scriptics.com/.

[Tuf82] E.R. Tufte. *The Visual Display of Quantitative Information*. Graphics Press, 1982.

[War94] M.O. Ward. Xmdvtool: Integrating multiple methods for visualizing multivariate data. *Proc. of Visualization '94, p. 326-33*, 1994.

[WB96] P.C. Wong and R.D. Bergeron. Multiresolution multidimensional wavelet brushing. *Proc. of Visualization '96, p. 141-8*, 1996.

[Weg90] E.J. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association, Vol. 411(85), p. 664*, 1990.

[Wil98] G.J. Wills. An interactive view for hierarchical clustering. *Proc. of Information Visualization '98, p. 26-31*, 1998.

[WL97] E.J. Wegman and Q. Luo. High dimensional clustering using parallel coordinates and the grand tour. *Computing Science and Statistics, Vol. 28, p. 361-8.*, 1997.

[WLT96] M.O. Ward, J.T. LeBlanc, and R. Tipnis. N-land: a graphical tool for exploring n-dimensional data. *CG194 Proc: Insight Through Computer Graphics, p. 130-41*, 1996.

[Wri95] W. Wright. Information animation applications in the capital markets. *Proc. Int'l Symp. Information Visualization*, pages 19–25, 1995.

[WTPea95] J. A. Wise, J. J. Thomas, K. Pennock, and et al. Visualizing the non-visual: spatial analysis and interaction with information from text documents. *InfoVis'95, p. 51-58*, 1995.

[WvL93] J. J. Wijk and R. D. van Liere. Hyperslice. *Proc. Visualization '93*, pages 119–125, 1993.

[WYR00] M. O. Ward, J. Yang, and E. A. Rundensteiner. Hierarchical exploration of large multivariate data spaces. In *Proc. Dagstuhl Seminar on Scientific Visualization*, 2000.

[Yan03] J. Yang. *Visual Hierarchical Subspace Indexing (VHSI) for High Dimensional Datasets (to be submitted).* PhD thesis, Dept. of Computer Science, Worcester Polytechnic Institute, 2003.

[YPWR03] J. Yang, W. Peng, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. *IEEE Symposium on Information Visualization*, October 2003.

[YWR02] J. Yang, M. O. Ward, and E. A. Rundensteiner. Interactive hierarchical displays: A general framework for visualization and exploration of large multivariate data sets. *Computer and Graphics Journal*, 27:256–283, 2002.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record, vol.25(2), p. 103-14*, June 1996.