A General Framework for Multi-Resolution Visualization

by

Jing Yang

A Dissertation

Submitted to the Faculty of the

WORCESTER POLYTECHNIC INSTITUTE

In partial fulfillment of the requirements for the

Degree of Ph.D. in

Computer Science

by

____

April 2005

APPROVED:

____

Professor Matthew Ward, Dissertation Advisor

____

Professor Elke Rundensteiner, Dissertation Co-advisor

____

Professor David Brown, Committee Member

____

Professor Daniel Keim, External Committee Member

____

Professor Michael Gennert, Head of Department

Abstract

Multi-resolution visualization (MRV) systems are widely used for handling large amounts of information. These systems look different but they share many common features. The visualization research community lacks a general framework that summarizes the common features among the wide variety of MRV systems in order to help in MRV system design, analysis, and enhancement. This dissertation proposes such a general framework.

This framework is based on the definition that a MRV system is a visualization system that visually represents perceptions in different levels of detail and allows users to interactively navigate among the representations. The visual representations of a perception are called a view. The framework is composed of two essential components: view simulation and interactive visualization.

View simulation means that an MRV system simulates views of non-existing perceptions through simplification on the data structure or the graphics generation process. This is needed when the perceptions provided to the MRV system are not at the user's desired level of detail. The framework identifies classes of view simulation approaches and describes them in terms of simplification operators and operands (spaces). The simplification operators are further divided into four categories, namely sampling operators, aggregation operators, approximation operators, and generalization operators. Techniques in these categories are listed and illustrated via examples. The simplification operands (spaces) are also further divided into categories, namely data space and visualization space. How different simplification operators are applied to these spaces is also illustrated using examples.

Interactive visualization means that an MRV system visually presents the views to users and allows users to interactively navigate among different views or within one view.

Three types of MRV interface, namely the zoomable interface, the overview + context interface, and the focus + detail interface, are presented with examples. Common interaction tools used in MRV systems, such as zooming and panning, selection, distortion, overlap reduction, previewing, and dynamic simplification are also presented.

A large amount of existing MRV systems are used as examples in this dissertation, including several MRV systems developed by the author based on the general framework. In addition, a case study that analyzes and suggests possible improvements for an existing MRV system is described. These examples and the case study reveal that the framework covers the common features of a wide variety of existing MRV systems, and helps users analyze and improve existing MRV systems as well as design new MRV systems.

Acknowledgements

I would like to express my greatest gratitude to my advisors, Matt and Elke, for their guidance, inspiration, and help to me. I would like to gracefully thank Prof. Brown and Dr. Keim for their valuable contributions to this work. I'd also like to thank my family for their support of me.

This work was funded by the NSF under grants IIS-0119276 and IIS-9732897.

Publications

Aspects of the work describe in this dissertation feature in the following publications:

- J. Yang, A. Patro, S. Huang, N. Mehta, M. Ward and E. Rundensteiner: Value and Relation Display for Interactive Exploration of High Dimensional Datasets. *Proc. InfoVis 2004*: 73-80
- M. Ward and J. Yang: Interaction Spaces in Data and Information Visualization. *Proc. VisSym 2004*: 137-146
- J. Yang, M. Ward and E. Rundensteiner: Interactive Hierarchical Displays: A General Framework for Visualization and Exploration of Large Multivariate Data Sets. *Computers & Graphics 27(2)*: 265-283 (2003)
- J. Yang, M. Ward, E. Rundensteiner and A. Patro: InterRing: A Visual Interface for Navigating and Manipulating Hierarchies. *Information Visualization 2(1)*: 16-30 (2003)
- J. Yang, W. Peng, M. Ward and E. Rundensteiner: Interactive Hierarchical Dimension Ordering, Spacing and Filtering for Exploration of High Dimensional Datasets. *Proc. InfoVis 2003*: 105-112
- J. Yang, M. Ward, E. Rundensteiner and S. Huang: Visual Hierarchical Dimension Reduction for Exploration of High Dimensional Datasets. *Proc. VisSym 2003*: 19-28
- J. Yang, M. Ward and E. Rundensteiner: Hierarchical Exploration of Large Multivariate Data Sets. *Data Visualization: The State of the Art* 2003: 201-212
- J. Yang, M. Ward and E. Rundensteiner: InterRing: An Interactive Tool for Visually Navigating and Manipulating Hierarchical Structures. *Proc. InfoVis 2002*: 77-84
- E. Rundensteiner, M. Ward, J. Yang, and P. Doshi, ”XmdvTool: Visual Interactive Data Exploration and Trend Discovery of High-dimensional Data Sets”, *Proc. SIGMOD 2002*: 631

Contents

List of Figures

Chapter 1

Introduction

1.1 An Introduction to Multi-Resolution Visualization Systems

Before providing a formal definition of a multi-resolution visualization system, I temporarily define it as a visualization system that presents data at multiple levels of detail. I will illustrate why the multi-resolution techniques are needed, and how the multi-resolution techniques are used to solve problems using examples from a variety of visualization applications.

Example 1: Multi-resolution Techniques in Video-on-Demand Applications

Video-on-Demand (VOD) is a critical technology for many important multimedia applications such as distance learning and electronic commerce [LLG99]. VOD allows remote users to playback any videos from a large collection of videos stored in one or more video servers [HCS98]. VOD services are different from traditional television broadcasting services in that users subscribing to VOD services have more control capabilities. For example, they can tailor the quality of video playback, and use VCR operations such as fast-forward or fast-rewind. [LLG99] presents a design of a cost-efficient VOD storage server that uses multi-resolution techniques.

The utilization of multi-dimensional techniques in [LLG99] was motivated by the following problems in VOD applications:

- Different users may desire different qualities of video playback (different video resolutions) according to their available display station and network bandwidth.

- VCR operations such as fast-forward and fast-rewind may require the server to retrieve more information at the same time period if only a single resolution is available. Thus additional I/O, network bandwidth and system buffer space may be demanded if there is only one resolution available.

Multi-resolution techniques are used to solve these problems. For example, [LLG99] uses sub-band coding techniques to design and implement a VOD storage server. Sub-band coding techniques [TZ94, Ohm94] are a class of scalable video compression algorithms capable of producing multiple resolutions of a video bit stream. They are based on multi-stage algorithms, where each stage uses quadrature mirror filtering techniques to separate the input stream into two frequency spectrums, one for the high-band and the other for the low-band frequency spectrum. Figure 1.1 shows an example of a two stage encoding scheme. The first filter decomposes the original video stream into a high-resolution component and another component. The second filter then further decomposes the other component into a medium-resolution component and a low-resolution component. The low-resolution video is obtained by directly using the decoded low-resolution component. The medium-resolution is obtained by adding the decoded low-resolution and medium-resolution components. The high-resolution video is obtained by adding the low, medium, and high decoded resolution components together.

Using sub-band coding techniques, the server proposed in [LLG99] is able to maintain

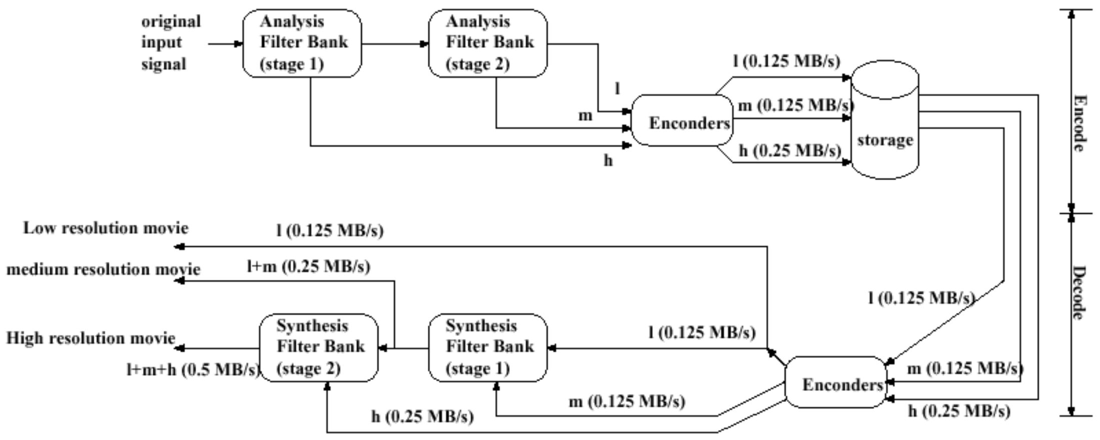


Figure 1.1: Illustration for sub-band coding technique for three resolution components [LLG99]. This figure is used without the authors' permission.

the same bandwidth requirements on disks and networks during both the normal and VCR displays by using video streams of different resolutions, and provide users with video of different resolutions.

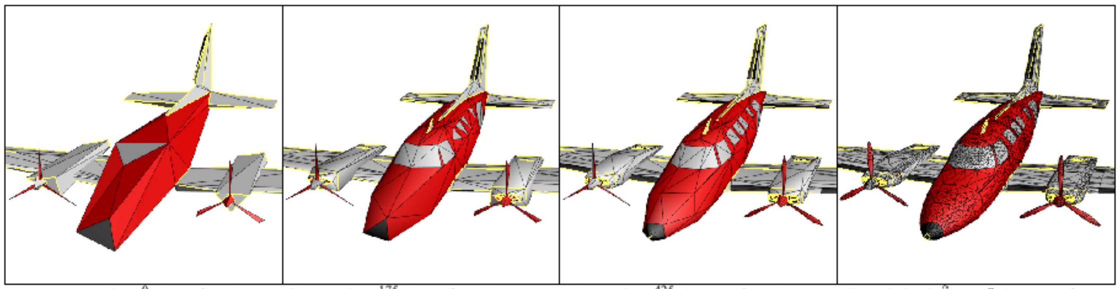
Example 2: Multi-Resolution Techniques in Internet-Based Server 3D Applications

Internet-Based Server 3D (IBS 3D) applications require fast transmission of a geometric model from the server to the client and fast rendering at the client. Multi-resolution techniques, such as the progressive mesh coding techniques proposed in [Hop96, KBG02], are widely used in IBS 3D applications.

The motivation of using multi-resolution techniques in IBS 3D applications is that due to the limit of network bandwidth, a high-resolution version of a large geometric model may not be transmitted to the client on time in an interactive mode. Multi-resolution techniques are used to solve this problem.

Progressive mesh representation [Hop96, KBG02] is a typical multi-resolution technique used in this field. It is a scheme for storing and transmitting meshes. In the progressive mesh form, an arbitrary mesh can be expressed as a much coarser mesh together with sequential detail records that indicate how to incrementally refine the coarser mesh exactly back into the original mesh [Hop96]. It allows the server to reconstruct a rough resolution of the mesh after a small number of bits has been received, and to gradually refine the mesh as more bits arrive. Thus the best fidelity possible with the smallest number of bits could be achieved.

For example, Figure 1.2 [Hop96] shows a model built using meshes with different resolutions.



(a) Base mesh M^0 (150 faces) (b) Mesh M^{175} (500 faces) (c) Mesh M^{425} (1,000 faces) (d) Original $\hat{M}=M^n$ (13,546 faces)

Figure 1.2: The progressive mesh representation of an arbitrary mesh captures a continuous-resolution family of approximating meshes [Hop96]. This figure is used without the authors' permission.

Example 3: Time-Critical Multi-resolution Volume Rendering

It is important to control rendering time in time-critical volume rendering applications such as virtual surgery or real-time volume navigation. However, the speed of volume rendering is often not fast enough for large volume data sets. Multi-resolution techniques, such as hierarchical rendering methods, are used to tradeoff between the quality and speed when rendering such data sets (a lower resolution requires less rendering time).

For example, [LS02] presents a hierarchical rendering approach for time-critical volume rendering. This approach builds a hierarchical volume structure on a large volume data set. First, the entire volume space is divided into smaller blocks called sub-volumes. Then multiple resolutions are created for each sub-volume. A lower resolution representation is generated from a higher resolution representation by averaging every $2 * 2 * 2$ voxels. During the rendering stage, proper resolutions of the sub-volumes are selected using an automatic resolution selection algorithm to make sure that the rendering time is controlled precisely according to a user defined frame rate.

Example 4: Multi-Resolution Techniques in Flow Visualization

Flow visualizations with high resolutions are often so cluttered that it is difficult to see or understand any aspects of the flow. For example, Figure 1.3 shows a topological graph cluttered with the tangent curves [NJS98]. Lower resolution displays that eliminate the clutter of high order details and keep the major trends of the flow would help users find overview aspects of the flow field [NJS98]. Multi-resolution techniques are used in flow visualization to solve the clutter problem.

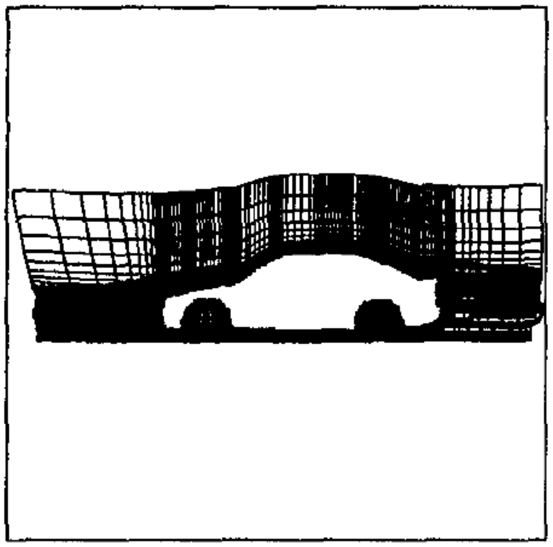


Figure 1.3: A topological graph cluttered with tangent curves [NJS98]. This figure is used without the authors' permission.

For example, [NJS98] presents a multi-resolution model for analyzing and visualizing two-dimensional flows over curvilinear grids. This model is based upon refinement and decomposition functions defined in a piecewise manner over a sequence of nested domains, which are generated by a vertex removal process. Lower (higher) resolutions of the flow field are generated by decomposition (refinement) functions from the higher (lower) resolutions. Haar wavelets and piecewise linear wavelets are used to generate the refinement and decomposition functions. The low resolutions present the major trends of the flow while eliminating high order details.

With this model, resolutions can be changed locally within the flow field. Thus efficient zoom in and zoom out can be achieved in this model with resolution changes. Zoom in can be performed with a resolution increase while zoom out can be performed with a resolution decrease. In addition, details can be added or eliminated at any location upon user requests. This is illustrated in Figure 1.4 [NJS98].

Example 5: Multi-Resolution Techniques for Supporting Mutual Awareness

Multi-resolution techniques can be used in applications where peripheral awareness is acquired through shared media spaces among people who are geographically dispersed but want to stay better in touch. For example, the AROMA project [PS97] was developed for such an application. In this project, rather than using the full original source of the signals, abstract representation of the source signals are used by removing data from the original signals. Different from other visualizations where multi-resolution techniques are used for effectiveness and efficiency, here the purpose of abstraction is psychological. For users who read the signals, the abstraction representation helps to reduce disturbance while still maintaining awareness. For users being watched by other users, the abstract representation provides a kind of shielding for privacy of them.

Motivation of Using Multi-Resolution Visualization

By checking “why” multi-resolution techniques are used in the above examples, it can be found that the purpose of using multi-resolution visualization techniques is solving the

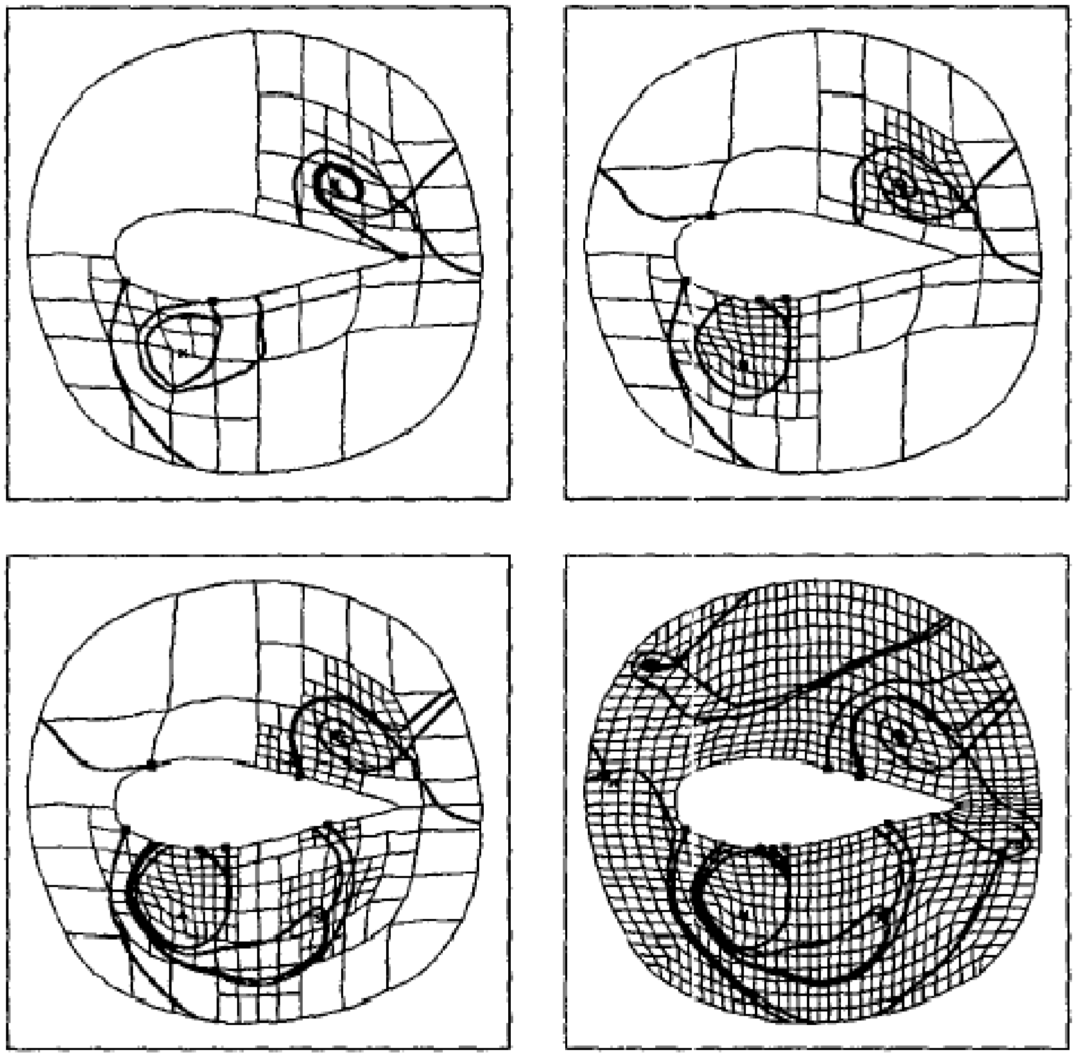


Figure 1.4: Zoom in and Zoom out in the multi-resolution flow visualization [NJS98]. This figure is used without the authors' permission.

following problems in visualization:

- limited computational resources;
- restricted cognitive resources of human viewers;
- social issues such as protecting data security, shielding human privacy, and keeping peripheral awareness.

Through these examples, it can be seen that a wide variety of multi-resolution techniques are used in all kinds of visualization fields. However, a framework that systematically summarizes how to design, analyze and evaluate multi-resolution visualization systems does not exist. Are such frameworks needed? According to my own research activities on multi-resolution visualization over the previous five years, I would say yes.

1.2 Motivation for Creating a General Framework for Multi-Resolution Visualization Systems

I have proposed and developed many different multi-resolution visualization approaches. However, this was not planned in advance. They were all triggered by a simple task of extending the hierarchical parallel coordinates approach to other multi-dimensional visualization techniques that was assigned to me five years ago. However, it turned out that this simple task gave birth to many new multi-resolution visualization techniques. This story is also the motivation for this dissertation.

Hierarchical Parallel Coordinates (HPC) [FWR99b] is a multi-resolution version of parallel coordinates [ID90, Weg90]. The main idea of HPC is to construct multiple views at different levels of detail for the same large data set by applying a hierarchical clustering algorithm on the data set. In a view at a low level of detail, similar data items are aggregated into clusters. Navigation and selection tools are used so that users can select clusters and display them in an extension of parallel coordinates. In this extension, means and extents of clusters are visually presented.

My task was to extend the hierarchical display from parallel coordinates to other three multi-dimensional visualization techniques, namely star glyphs [SFGF72], scatterplot matrices [CM88], and dimensional stacking [LWW90]. It was a very simple task – the way the data was abstracted and the navigation and selection tools did not need to change, thus I only needed to figure out how to visually convey cluster means and extents in the three displays. Soon three new multi-resolution visualization techniques were generated. They share the same method of data abstraction and graphical mapping, and the same interaction tools. Actually the common data abstraction, mapping, and interaction tools form a visualization framework named the Interactive Hierarchical Display (IHD) framework [YWR03b, YWR03a], and it can be easily applied to most existing multi-dimensional visualization techniques.

Then my advisor, Matt Ward, asked: if the data items can be clustered and visualized in a multi-resolution manner, can we cluster the dimensions and visualize them in a multi-resolution manner? I worked on it, and the answer was that we could. The data dimension space is an orthogonal space to the data item space, and the clustering operation can be applied to the dimensions in a similar way as it is applied to the data items. This idea led to the Visual Hierarchical Dimension Reduction (VHDR) approach [YWRH03, YPWR03] for interactive exploration of high dimensional data sets.

High dimensions cause clutter in traditional multi-dimensional visualizations. In VHDR, a dimension hierarchy is constructed upon all dimensions in a high dimensional data set using a hierarchical clustering algorithm. Similar dimensions form dimension clusters in the dimension hierarchy. Users can interactively navigate the dimension hierarchy and select dimension and dimension clusters of interest from it. The data items are then mapped to a subspace composed of the selected dimension and dimension clusters and visualized in traditional multi-dimensional visualization techniques. A dimension cluster is an aggregation of all dimensions composing the cluster, and is visualized as a single dimension in the subspace visualization with some extensions for visually conveying the dissimilarity of the dimensions within the cluster. Obviously, VHDR is also a multi-resolution approach. Similar to the IHD approach, hierarchical clustering is used in VHDR. However, clustering is applied upon the dimension space in VHDR, while in

IHD clustering is applied upon the data item space. Many interactive operations in IHD have been transplanted to the VHDR approach.

Then I asked a further question: if clustering can be applied upon the dimension space to generate multi-resolution visualizations, can Multi-Dimensional Scaling (MDS) [KW78, Dav83, Mea92, CC94] also be applied to the dimension space to generate multi-resolution visualization? The reason is that MDS is often applied to the data item space [WTP$^+$95] and it can convey the data item relationships more accurately than the clustering approach. If clustering can be transplanted from the data item space to the dimensions space, why not MDS?

This idea led to another new multi-resolution visualization approach to visualizing high dimensional data set. It is named Value and Relation (VaR) display [YPH$^+$04]. In a VaR display, a proximity positioning of the dimensions is generated using MDS. Each dimension is represented by a pixel-oriented glyph that explicitly conveys the data values in the dimension. Closely related dimensions are positioned close to each other while unrelated dimensions are positioned far away from each other. Users can view the relationships among a subset of dimensions in more detail by applying MDS on a selected subset of dimensions to refine the proximity positioning. Users can also examine data values in more detail by enlarging the sizes of the dimension glyphs. Case studies [YPH$^+$04] have shown that the VaR displays can effectively visualize real data sets containing more than three hundred dimensions. It is an achievement since most traditional multi-dimensional visualization techniques clutter when there are tens of dimensions. The VaR displays provide a rich set of navigation and selection tools, most of which are transplanted from the previous IHD and VHDR approaches.

The IHD, VHDR, and VaR approaches are all multi-resolution visualization approaches. It was amazing how easily it was to generate a new multi-resolution visualization by borrowing ideas and tools from existing ones. For example, the IHD framework used the same abstraction approach (clustering) on the same space (the data item space) and the same interaction tool set as the hierarchical parallel coordinates display, and simply replaced the visualization method of parallel coordinates by other methods such as star glyphs, scatterplot matrices, and dimensional stacking. The VHDR approach used the same abstraction approach (clustering) as the IHD approach, but applied it upon an different space (the dimension space). The VaR approach used a different abstraction approach (MDS) and applied it upon the same space (the dimension space) as the VHDR approach. All these approaches share many interaction tools.

The above story gave me a hint that there are some essential features shared by various multi-resolution visualization systems. If these features can be explicitly identified, generation, analysis, and evaluation of multi-resolution visualization systems can perhaps be much easier. By studying a wide variety of multi-resolution visualization systems and multi-resolution models, I reached a general framework for multi-resolution visualization systems and present it in this dissertation. Since my research focus in the past five years was information visualization, most of the visualization systems I analyzed for generating the framework and the examples used for illustrating the framework are in the scope of information visualization. Thus this framework will be most useful for designing, analyzing, and evaluating multi-resolution visualization systems in information visualization. However, visualization systems and examples from other visualization fields are also involved in the dissertation. I believe that the proposed framework is also a useful reference for designing, analyzing, and evaluating multi-resolution visualization systems beyond information visualization.

The following chapters of this dissertation are organized as follows: Chapter 2 proposes a semantic definition of multi-resolution visualization systems and gives an overview of the general framework. Chapter 3 to 7 discuss the framework in details. Chapter 7 presents a case study that uses the framework to analyze and improve an existing visualization. Chapter 8 concludes with a summary and contributions of this dissertation as well as potential directions for future research. A multitude of examples are used throughout the dissertation to convey the concepts of the framework.

Chapter 2

Semantics of the Framework

In this chapter, I first propose a formal definition of multi-resolution visualization systems, which originated from an abstraction model [SZ98] developed in Artificial Intelligence. This formal definition summarizes the essential features of multi-resolution visualization systems. Then, I propose a general framework for multi-resolution visualization systems based on the formal definition.

This chapter is structured as follows: Section 2.1 introduces the abstraction model and other related work on the formal definition and framework. Section 2.2 gives the formal definition of multi-resolution visualization systems, while Section 2.3 presents a brief description of the general framework. The framework will be discussed in more detail in the following chapters.

2.1 Related Work

[SZ98] presents a semantic abstraction model for concept representation and learning. It was based on the observation that a conceptualization of a domain involves entities belonging to at least three levels. The first level, which is the fundamental level, is the *perception of the world* (where concrete objects reside). The second level is the *memorization of the perception*, where some kinds of structure are used to describe objects and relations perceived in the world. The third level is the *description of the memories*, where a language is used to communicate with others, and to perform reasoning on the memorized structures. In this model, *abstraction* is understood as a mapping between views of the world, rather than a mapping between structures or languages. The modifications of the structures and the languages, which is called *simplification*, are viewed as side effects that are necessary for describing what happens at the level of the perceived world. A set of operators is defined to realize the abstraction process. This model was referred to as the KRA model in some references. Since visualization techniques can be viewed as visual languages that describe information, and visual exploration of data sets can be viewed as a kind of reasoning through interactive visualization, it seems plausible that the KRA model can be used to analyze multi-resolution visualization systems. Figure 2.1 gives an illustration of the KRA model.

[MZS99] uses the KRA model to detangle the process of representation (change of language) from the process of abstraction (change of level of detail) in cartography. These two processes are usually very much entangled in cartography. [MZS99] claims that according to the KRA model, knowledge abstraction in cartographic generalization is the identification of abstracted geographic objects relevant to the user needs, while knowledge representation is the process of symbolizing abstracted objects. I will use similar analyses to detangle abstraction from interactive visualization in the general framework for multi-resolution visualization systems.

[Krü98] presents a framework that uses three abstraction layers to summarize graphical simplification techniques used in 3D volume visualization. This framework is based on a three level rendering pipeline (model space, generation space, and image space) model. The three abstraction layers are model space, generation space, and image space.

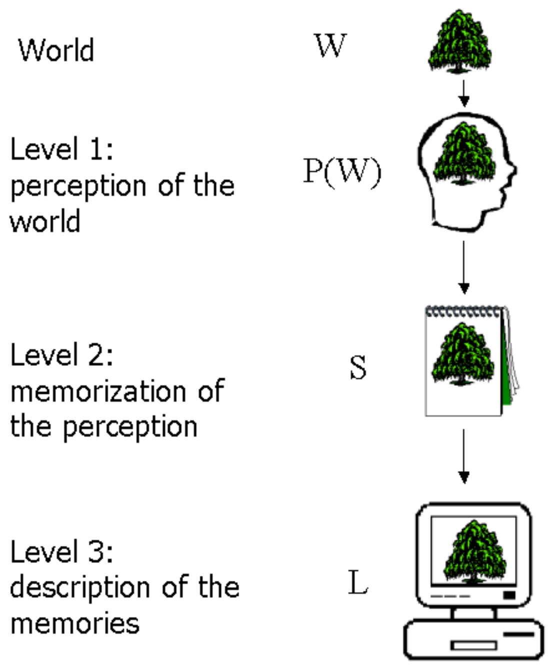

World
W
Level 1: perception of the world
P(W)
Level 2: memorization of the perception
S
Level 3: description of the memories
L

Figure 2.1: Illustration for the KRA model. The world W where a concrete tree exists is observed. The observer perceives of the tree and has a perception P(W) in his mind. He records the perception P(W) in a structure S. The recorded structure S is then visualized using a language L.

In the model space, simplification techniques aim at reducing the representation complexity of 3D models. An example is to merge primitive elements such as cylinders and cubes. In the generation space, simplification techniques such as suppression of colors can be used in order to simplify the depictions of 3D models. In the image space, simplification techniques are used to manipulate the images directly, including a variety of filters or erasers. The discussion of this paper is restricted to 3D volume visualization. However, it reveals that simplification can happen in all levels of the rendering pipeline.

[WY04] identifies unifying themes and frameworks for various interaction techniques. It presents a list of spaces within which interactive operations can occur in information visualization. These spaces include screen-space (pixels), data value-space (multivariate data values), data structure-space (components of data organization), attribute-space (components of graphical entities), object-space (3D surfaces), and visualization structure-space. After defining a set of interaction operators, interaction techniques can be viewed as combinations of the interaction operators and the spaces. Although the purpose of [WY04] is to analyze interaction techniques, such a classification of spaces can also help to analyze simplification by viewing simplification techniques as combinations of simplification operators and spaces.

2.2 Formal Definition of Multi-Resolution Visualization Systems

2.2.1 Abstraction and Level of Detail

Definition: *reasoning context* R = (P(W), S, L) [SZ98]

World W: W is the world where the objects reside.

Perception P(W): P(W) is the perception that an observer has of the world. The perception exists only for the observer and only during its being perceived.

Structure S: S is an extensional representation of the perceived world, in which stimuli relating one to another are stored together.

Language L: L is a language that allows reasoning about the perceived world and communication with others. L allows the perceived world to be described intentionally.

In the example shown in Figure 2.1, P(W), S, and L is defined.

Definition: Given a world W, let $P_1(W)$ and $P_2(W)$ be two perception systems for W, and let $\Gamma 1$ and $\Gamma 2$ be their corresponding configuration (stimuli) sets. The perception system $P_2(W)$ will be said to be simpler than P1(W) iff:

- There is a function f: $\Gamma 1 \rightarrow \Gamma 2$ that starting from any configuration $\gamma 1 \in \Gamma 1$ uniquely determines a configuration $\gamma 2 \in \Gamma 2$.

- The inverse function f^{-1} does not exist, because there is at least some configuration in $\Gamma 1$ that cannot be uniquely reconstructed from $\Gamma 2$, without additional information [SZ98].

Definition: Given a world W, let $R_g = (P_g(W), S_g, L_g)$ and $R_a = (P_a(W), S_a, L_a)$ be two reasoning contexts, which we conventionally label as *ground* and *abstract*. An *abstraction* is a functional mapping A: $P_g(W) \rightarrow P_a(W)$ between a perception $P_g(W)$ and a simpler perception $P_a(W)$ of the same world W [SZ98]. $P_g(W)$ is said to have a higher *level of detail* than $P_a(W)$.

According to this definition, abstraction is a mapping between perception of the world, rather than a mapping between structures or languages. The modification of the structures and of the language is only introduced to describe the differences occurred in the perception of the world, and is a *simplification*.

Definition: Given a world W, let $R_g = (P_g(W), S_g, L_g)$ and $R_a = (P_a(W), S_a, L_a)$ be two reasoning contexts. A simplification is a functional mapping B: $S_g \rightarrow S_a$, or a functional mapping C: $L_g \rightarrow L_a$, if there is an abstraction from $P_g(W)$ to $P_a(W)$ [SZ98].

Figure 2.2 gives an illustration of abstraction and simplification.

2.2.2 Abstraction Hierarchy

According to [SS77], the objective of abstraction is to allow users to heed details of the world that are relevant to the reasoning and to ignore other details. There may exist too many relevant details for a single abstraction to be intellectually manageable. Such manageability can be provided by using an abstraction hierarchy, which is defined as:

Definition: Given a world W, let $R_1 = (P_1(W), S_1, L_1)$, $R_2 = (P_2(W), S_2, L_2)$, ..., and $R_k = (P_k(W), S_k, L_k)$ be k reasoning contexts. $P_1(W), P_2(W), ..., and P_k(W)$ form an abstraction hierarchy with k levels of detail, if there is a functional mapping A:

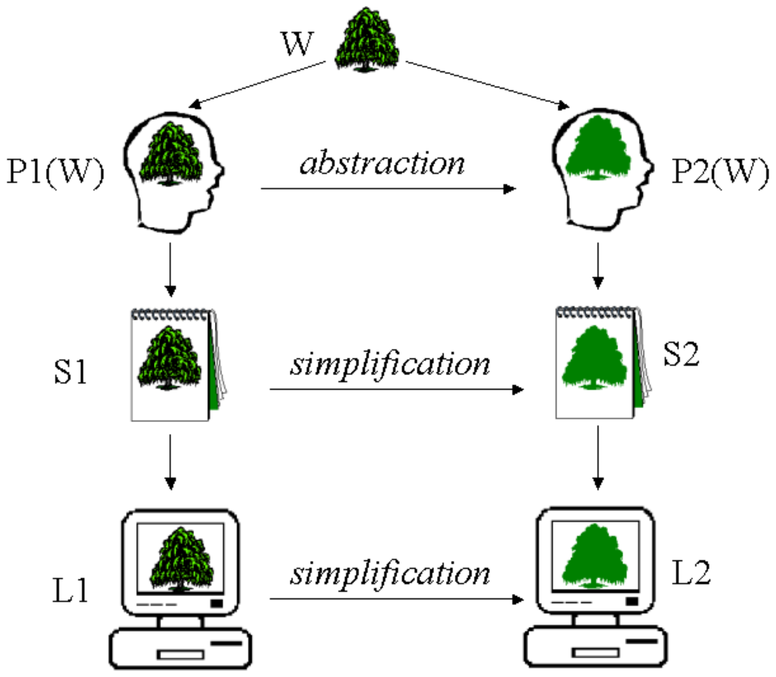


Figure 2.2: Illustration for abstraction and simplification. P1(W) and P2(W) are two perceptions of the same world W. P2(W) is simpler than P1(W). S1 and S2 are the structures memorizing P1(W) and P2(W). L1 and L2 are the languages describing S1 and S2. The mapping from P1(W) to P2(W) is called abstraction, while the mappings from S1 to S2, and from L1 to L2 are called simplification.

$P_i(W) \rightarrow P_{i+1}(W)$ between a perception $P_i(W)$ and a simpler perception $P_{i+1}(W)$ of the same world W for each $1 <= i <= k - 1$.

An abstraction hierarchy allows relevant details to be introduced in a controlled manner. Many relevant details in a view on a given level in the hierarchy can be temporarily ignored when understanding its abstraction on the next level. Multi-resolution visualization systems visually convey views in abstraction hierarchies.

2.2.3 Definition of Multi-Resolution Visualization Systems

Visualization is a visual reasoning activity. The previous definition of abstraction hierarchy can be applied to visualization to get the definition of multi-resolution visualization systems:

Definition: A *multi-resolution visualization system* is a visualization system that visually represents abstraction hierarchies of perceptions and allows users to interactively navigate within and among the visual representations.

All visual representations for a perception are called a *view*. For example, in Figures 2.1 and 2.2, views are the trees displayed in the computers. Please note that the definition of a view in this framework is different from that in the database literature, where a view is defined as a virtual table in the relational data model.

A view has the same level of detail as the perception it represents. Through a multi-resolution visualization system, users can learn global information and locate relevant details quickly from views at low levels of detail, and examine relevant details from views at high levels of detail.

A multi-resolution visualization system can be linked with the KRA model. In the context of multi-resolution visualization, structure S is a data set to be visualized, while L is the visual language conveying the data sets to the users, namely the process of generating graphics from the data sets.

It is ideal for multi-resolution visualization systems if the world is perceived in user desired levels of detail. However, in practice, it often happens that the world is perceived in fewer levels than that is desired by the users. In such cases, multi-resolution visualization systems have to simulate views through simplification on the structure level S and/or language level L, in other words, simplification on the data structure and/or the graphic generation process. Figure 2.3 gives an example of how a view is simulated through simplification in the structure level when the perception it represents does not really exist. Figure 2.4 gives an example of how a view is simulated through simplification in the language level when the perception it represents does not really exist.

Taking the above definition and considerations into account, I propose a general

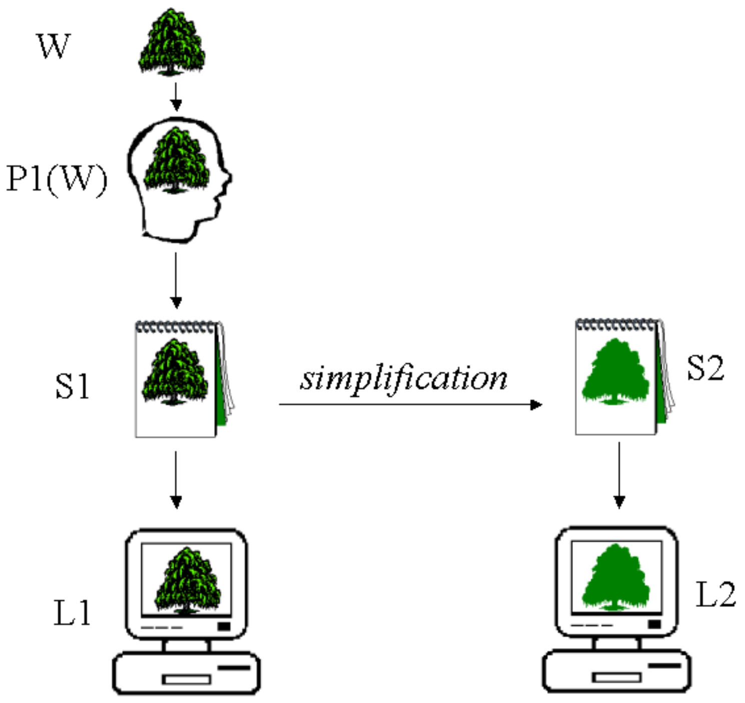


Figure 2.3: Illustration of view simulation by simplification in the structure level. The view in the second computer is simulated. The perception it represents does not really exist.

framework for multi-resolution visualization systems in the following section.

2.3 General Framework for Multi-Resolution Visualization Systems

According to the definition of a multi-resolution visualization system presented in Section 2.2, the essential feature of a multi-resolution visualization system is that it represents abstraction hierarchies of perceptions and allows users to interactively navigate through the views associated with them. Since perceptions provided by the observers are often not in a hierarchy of a user's desired levels of detail, a multi-resolution visualization system often needs to simulate views through simplification in the structure level or the language

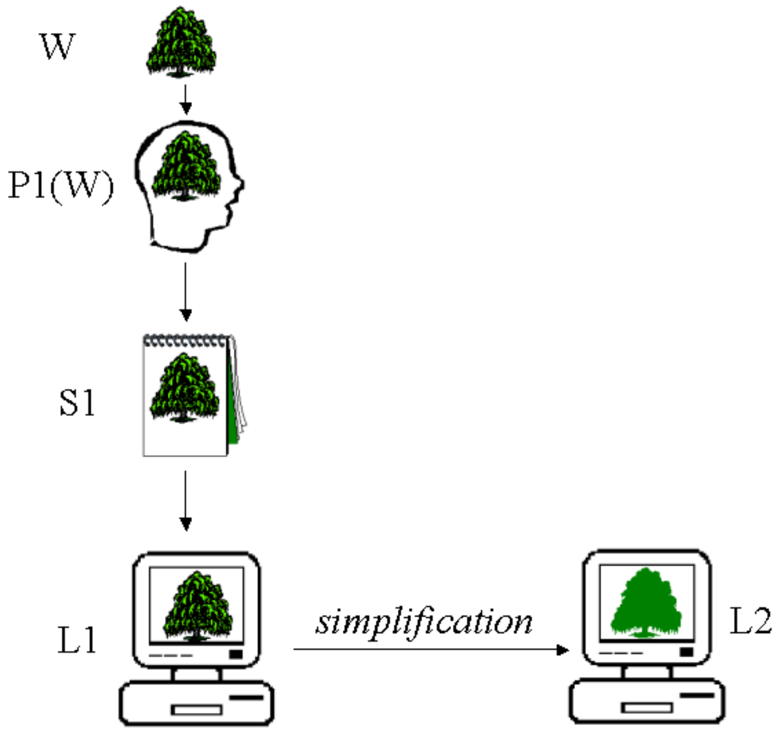


Figure 2.4: Illustration of view simulation by simplification in the language level. The view in the second computer is simulated. The perception it represents does not really exist.

level. I propose a general framework for multi-resolution visualization systems. It is composed of two essential components:

View Simulation This stage simulates views through simplification at the structure level (data sets) or the language level (graphic generation processes) if desired perceptions are not provided.

Interactive visualization This stage visually presents the views to users and allows users to interactively navigate among different views or within one view.

A wide variety of view simulation approaches has been used in existing multi-resolution visualization systems. While some of the approaches might appear quite unrelated, they actually may share a number of common features. I try to identify classes of view simulation approaches and describe them in terms of simplification operators and operands (the spaces upon which the operators are applied). This approach is inspired by our work on identifying and describing the wide variety of interaction techniques [WY04]. In Chapter 3, I present a list of simplification operators such as sampling operators, aggregation operators, and approximation operators. In Chapter 4, I present a list of spaces, such as data space and visualization space.

There is also a wide variety of ways to present the views and interaction tools in the interactive visualization stage. In Chapter 5 I provide a list of approaches to presenting multiple views to users, such as the overview + detail interface, the focus + context interface, and the zoomable user interface, as well as common user interactions provided by multi-resolution visualization systems.

Chapter 3

Simplification Operators

In this section, I attempt to categorize the wide range of simplification operators that can be used in multi-resolution visualization. Some of them are commonly used in existing multi-resolution visualization systems, while some are not used in any multi-resolution visualization systems to date but are widely studied as simplification approaches in the database management field and other fields. I believe that they can be used in visualization and lead to new multi-resolution visualization approaches.

I divide the simplification operators into four categories, namely sampling operators, aggregation operators, approximation operators, and generalization operators. Each of them can be further divided into smaller categories. I introduce each of the wide categories and their sub-categories in the following sections. In order to illustrate an operator, I first give a general definition of it, and then give detailed description of some distinct techniques falling into the general definition. If possible, I give examples of visualization systems that use those techniques to enable multi-resolution visualization.

3.1 Sampling Operator

Definition: *sampling* - the process of selecting some part of a population to observe so that one may estimate something about the whole population [Tho92]. Sampling is an approach whose input is an object set $A = (a_1, a_2, ..., a_k)$ and whose output is another object set $B = (a_{i_1}, a_{i_2}, ..., a_{i_m})$, where $1 <= i_j <= k$, for $1 <= j <= m < k$.

Sampling techniques have been widely studied in the statistics and database management fields and numerous sampling techniques have been developed. However, only a few simple sampling techniques, such as simple random sampling, are widely used in multi-resolution visualization. These techniques are simple to use, but they are often not the best for visualization applications with regard to effectiveness and efficiency. In the following sections, I first briefly introduce simple random sampling, and then introduce some more complex sampling techniques that can be used for more effective and efficient multi-resolution visualization.

Simple Random Sampling

Definition: *simple random sampling* - a sampling procedure that assures that each element in the population has an equal chance of being selected.

Simple random sampling is widely used in multi-resolution visualization systems. For example, [DE02] proposed a 2D zooming interface named the Astral Telescope Visualiser. It is a 2D scatterplot where data items displayed are randomly sampled. It has a zoom in function such that the sampling rate increases when the display is zoomed in.

Multi-Level Sampling

In a database, a data set is often stored in many pages, and each page is composed of many rows. In order to get a sample of rows from the data, a *bi-level Bernoulli sampling* can be used. It is a sampling method that combines row-level and page-level sampling: for a given sampling rate q, it samples pages at a rate p ($p \geq q$). For each sampled page, it samples rows at a rate $r = q/p$ [HK04].

This definition can be extended to a definition of *multi-level sampling*: suppose that a data set is stored in a hierarchy and each node in the hierarchy is composed of multiple nodes in lower levels in the hierarchy except the leaf nodes. A sample of leaf nodes of this hierarchy can be generated in such a way: first sample the nodes in the second level of the hierarchy (the first level is the root node), then iteratively sample the children of the sampled nodes until leaf nodes are reached.

The bi-level Bernoulli sampling is a special case of multi-level sampling where the hierarchy has three levels. The root node is the data set, nodes in the second level are pages, and nodes in the third level are rows. I will only discuss the bi-level Bernoulli sampling since similar discussion can be easily extended to other multi-level sampling techniques.

The bi-level Bernoulli sampling combines the row-level and page-level Bernoulli sampling methods. The purpose is to trade off processing speed and statistical precision. Row-level sampling yields accurate results, but it is often expensive in I/O cost. Page-level sampling is much faster, but it often yields much less precise results. The bi-level sampling method combines row-level and page-level sampling: for a given sampling rate q, it samples pages at a rate p ($p >= q$). For each sampled page, it samples rows at a rate $r = q/p$. By adjusting p and r, the user can systematically trade off between processing speed and statistical precision. An algorithm has been proposed to adjust p and r [HK04] using the standard error of the unbiased estimator to measure the statistical precision and an I/O cost model to measure the processing speed.

Multi-level sampling can be applied to visualizations where the data set is stored in a hierarchy structure for multi-resolution visualization. There are multiple ways to do it. One approach is to directly visualize the sample generated and hide the multi-level sampling approach itself from the users. Users can change the total sampling rate to get views at different levels of detail. Another approach is to expose the multi-level sampling approach to the users, such as visually presenting the sampling rates at different levels of sampling to the users and allowing them to interactively change the rates. An advantage of this approach is that users can control the trade off between processing speed and statistical precision directly.

Biased Sampling

Biased sampling contains biases toward some objects in a population when generating samples. Biased sampling can be used for many different purposes. For example, in a visualization application, sampling can be biased to the objects in a data set that are more closely related to users' exploratory tasks, or biased to the objects that reflect the real world more accurately than others.

For example, [KGKB03] uses a biased sampling technique to speed up general data mining tasks such as clustering and outlier detection for large multidimensional data sets. The motivation of [KGKB03] is that using simple random sampling, a very large sample is needed to guarantee that a certain fraction of a small cluster is included in the sample, which is necessary for detecting that cluster in the sample. The solution proposed by [KGKB03] is to change the sampling rate according to the local density of a data set.

Firstly, the density of a data set is estimated using a kernel density estimation technique. Then, the probability that a given point will be included in a sample is biased according to its local density. According to different analysis objectives, the probability can be biased in different ways:

- To identify clusters, the dense regions can be oversampled.

- To identify outliers, the sparse regions can be oversampled.

Experiments reported by [KGKB03] showed that a much larger uniform sample was required to achieve the same effects in clustering as a biased sample. The biased sampling also performed better in finding small clusters.

Here I list some other scenarios that biased sampling can be used in multi-resolution visualization:

- If a data set contains a number of unreliable objects (for example, data items that contain many missing values), biased sampling can be used so that reliable objects have larger chances to be visualized than unreliable objects.

- If a user finds and marks interesting objects in interactive exploration and wants to keep track of them (or some of them) in the following exploration, biased sampling can be used so that marked objects have larger chances to be visualized than unmarked objects.

- In a focus + context display, biased sampling can be used so that objects in the focus or close to the focus have larger chances to be visualized than objects that are far away from the focus.

Dynamic Sample Selection

Dynamic sample selection is an approach where the sample is dynamically constructed by combining a subset of samples built during the pre-processing phase. Dynamic sample selection provides high quality samples at the cost of pre-processing a pre-stored set of samples.

For example, [BCD03] presents a dynamic sample selection approach for approximate query processing. In this approach, a family of non-uniform samples is constructed during a pre-processing phase. During the query-processing phase, an appropriately biased sample for estimating the answer of a given query is dynamically constructed by combining a subset of samples built during the pre-processing phase.

The motivation of [BCD03] is that an appropriately biased sample provides good accuracy for a particular set of queries but not for the others. For different queries, the "bias" might be different. Thus a single biased sample is not suitable for ad hoc exploratory data analysis where queries might vary a lot. Increasing the sample size can improve the accuracy of the approximation, but it increases the query processing time. The solution proposed by [BCD03] is to pre-process the data set to build and store a set of biased samples. Thus in the query-processing phase, an appropriately biased sample for a given query can be constructed using a subset of these samples. In particular, a query is analyzed and decomposed into queries on a subset of the pre-stored biased samples. A final approximate answer is composed out of the results of these queries. In this way, the accuracy of the approximate answer is improved since the samples used are appropriately biased to the query. The query processing time is not increased, because although lots of samples are stored, only a few of them are used for a query and the size of each sample is not big. The cost is that more disk space is used to store the pre-processed samples. This cost is affordable for most applications.

[BCD03] proposes a technique named *small group sampling* as an instantiation of dynamic sample selection for aggregation queries. In small group sampling, an overall sample table is built by sampling the original table with a base sampling rate. Besides the overall sample table, a small group table is built for each eligible column. A small group table contains all rows belonging to small groups of that column. In runtime, a query is decomposed into queries to the overall sample table (for big groups) and to one or more small group tables (for small groups). The final answer is composed out of answers to these queries. The rationale behind this technique is that uniform sampling is unfair for small groups. Thus an approach without downsampling small groups will give a better answer than uniform sampling.

Experiments reported by [BCD03] showed that dynamic selection of appropriate portions of previously constructed samples provide more accurate approximate answers than static, non-adaptive usage of uniform or non-uniform samples.

Another example of dynamic sample selection is a multi-resolution visualization approach I proposed. It is aimed to visualize multi-dimensional data sets containing a large number of data items. In this approach, a hierarchy of sets of samples is constructed in the pre-processing stage. This hierarchy corresponds to a hierarchy upon the data items generated using a hierarchical clustering algorithm. Each sample in the sample hierarchy can be mapped to a cluster in the data item hierarchy and is the sample of the data items included in the cluster. Users can interactively select levels of detail and region of focus in the sample hierarchy and a dynamic sample is constructed according to their selection by combining the selected subset of samples into a sample and visualizing this sample. Figure 3.1 shows scatterplot matrices of the AAUP data set at different levels of detail generated using this dynamic sample selection approach. Figure 3.2 gives examples of focus + context displays generated using the dynamic sample selection approach where a region is displayed in a higher level of detail than the other regions. This is not feasible using a simple random sampling approach.

When to Use, and Advantage and Pitfalls of Sampling Operators

[DE02] suggests several visualization scenarios where it is proper to use sampling:

- Any visualization based on aggregate or summary statistics can use sampled data to give approximations, as well as interfaces with numerical data.

- Visualizations containing points or lines that could saturate the display can use

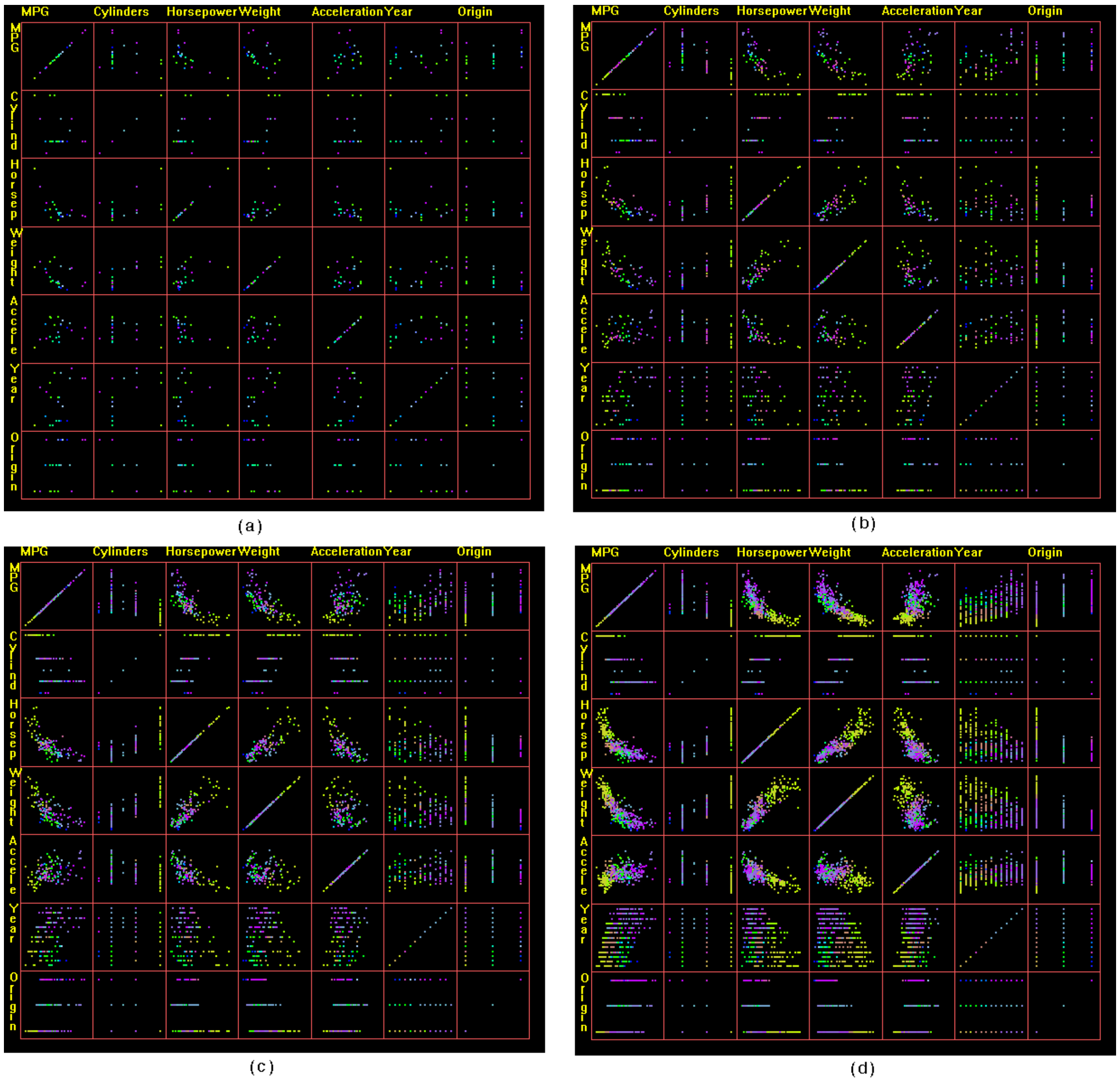


Figure 3.1: Scatterplot matrix displays of the AAUP data set in increasing levels of detail. They are generated using a dynamic sample selection approach.

sampled data to avoid saturation and reveal features and relationships.

- Sampling can be used to reduce the data set to a size that allows detailed visualization such as thumbnails of individual items.

The first two scenarios are easy to understand, so I only use an example to illustrate the third one. In the Informedia Digital Video Library application presented in [DCHW03], query results to the video library are represented using a collage of rep-

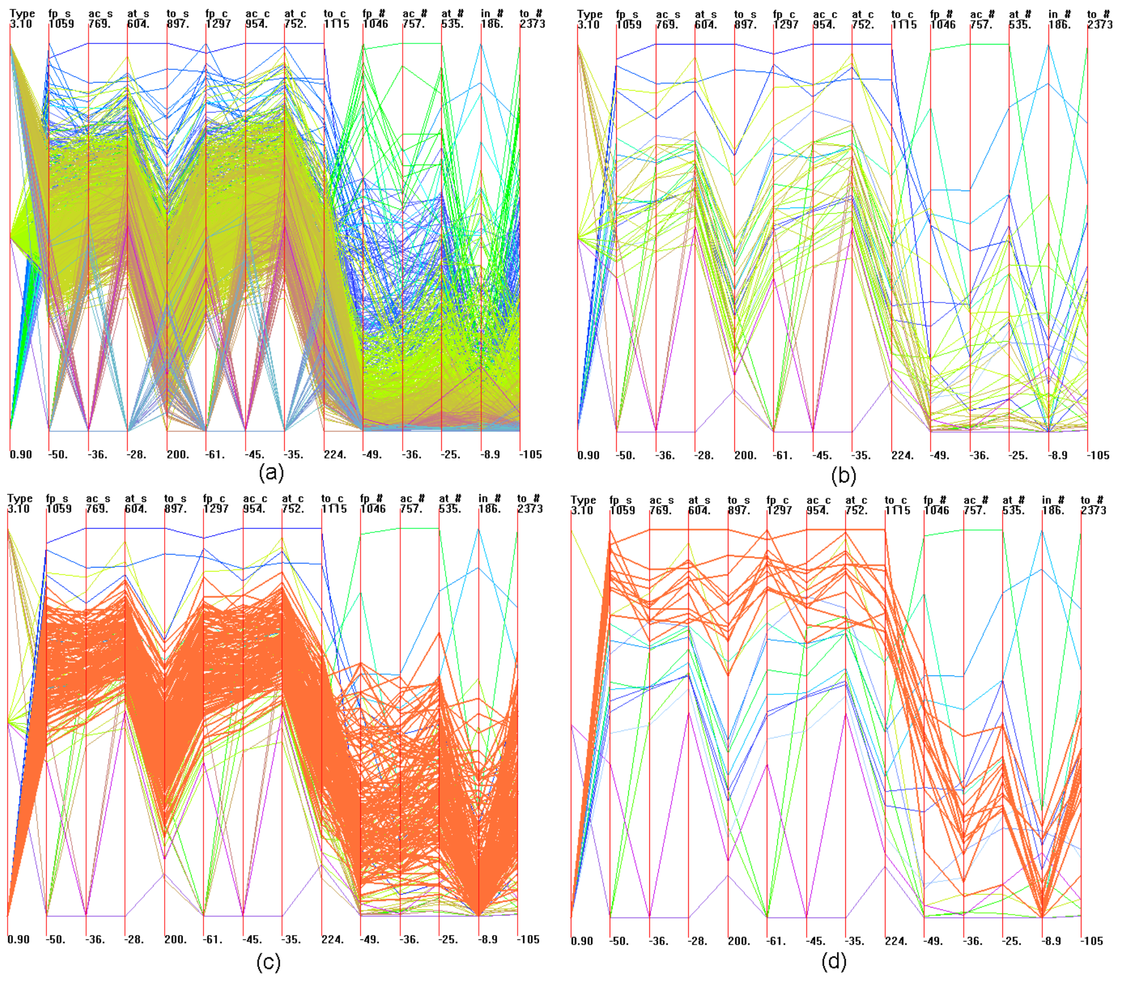


Figure 3.2: Parallel Coordinates displays of the AAUP data set generated using the dynamic sample selection approach. (a) The original data set (b) The data set in a low level of detail (c) A region of the data set is displayed in a higher level of detail than the other regions and highlighted. (c) Another region is displayed in a higher level of detail than the other regions and highlighted.

resentative keyframes. Occlusion can happen as shown in Figure 3.3. Sampling is used to reduce the number of keyframes shown in order to make sure that there is no occlusion in the display. Figure 3.4 shows the improved interface without occlusion.

Using sampling operators has several significant advantages. Firstly, the objects in the sample come from the original data set without any transformation or aggregation. Thus they have intuitive meaning to users. Similarly, the visual representation of the objects

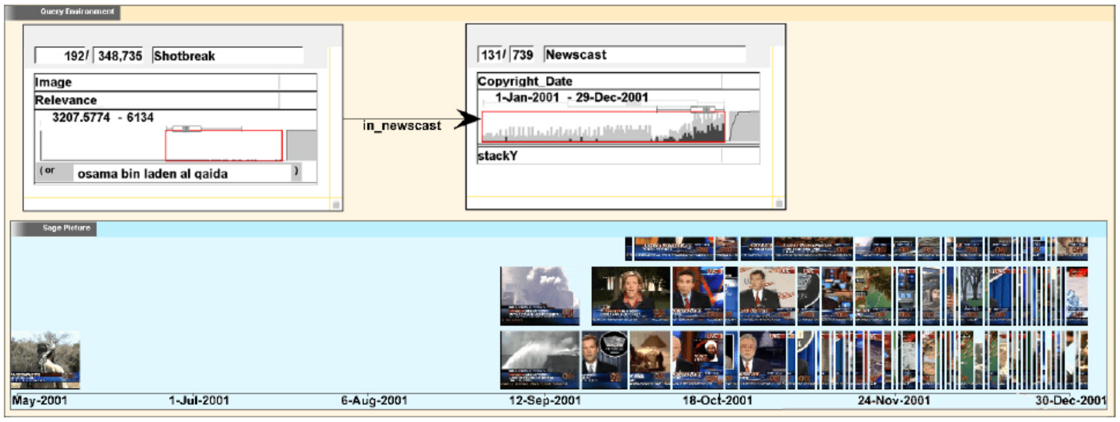


Figure 3.3: Query results to the video library are represented using a collage of representative keyframes [DCHW03]. Occlusion happens in this display. This figure is used without the authors' permission.

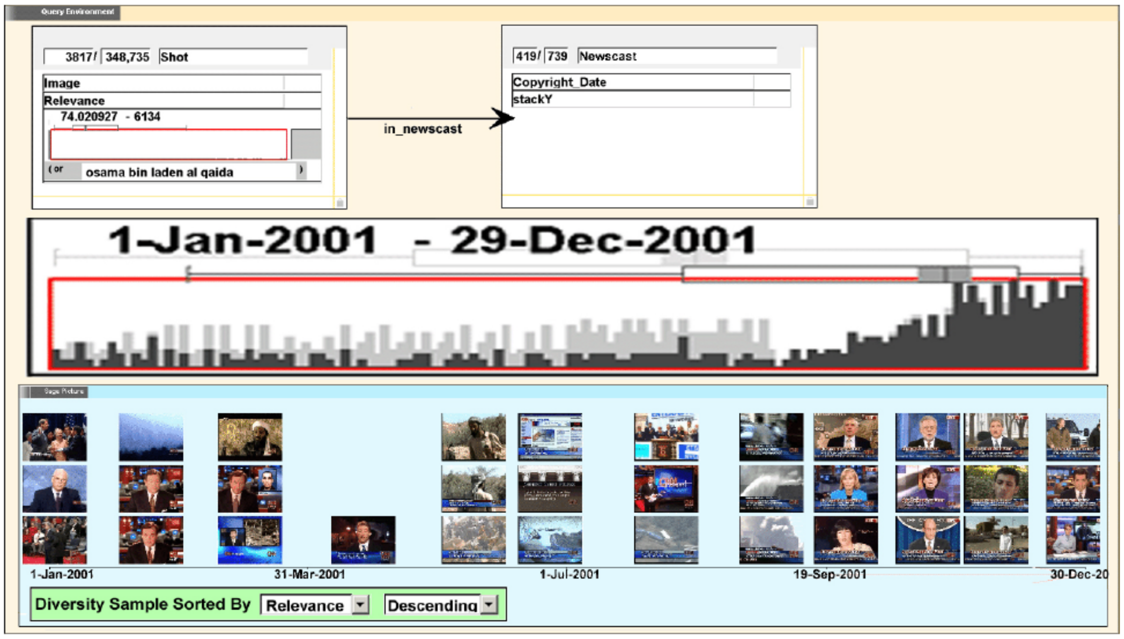


Figure 3.4: Sampling is used to reduce the number of keyframes shown in the display to make sure that there is no occlusion [DCHW03]. This figure is used without the authors' permission.

in the sample can be the same as the original objects. Thus the samples are ready to be visualized without any modification to the visual representation. Thirdly, some sampling techniques, such as random sampling, are economical. They need few pre-processing and auxiliary data structures. The samples can be generated on-line with little effort and the level of detail can be interactively changed easily.

However, it has been revealed that the information loss caused by sampling could be large with improper sampling techniques and sample sizes. For example, [KGKB03] illustrates how random sampling could lead to missing small clusters in the sample if the sample size is not big enough. In their example, D is a data set of size n, and u is a cluster of size $|u|$. According to [GRS98], a cluster u is included in the sample when more than $\phi * |u|$ points from the cluster are in the sample, for $0 <= \phi <= 1$. Then the sample size s, required by uniform random sampling, to guarantee that u is included in the sample with probability no less than δ, $0 <= \delta <= 1$ is:

$s >= \phi * n + \frac{n}{|u|} log(\frac{1}{\delta}) + \frac{n}{|u|}\sqrt{(log(\frac{1}{\delta}))^2 + 2 * \phi * |u| * log(\frac{1}{\delta})}$ (1)

[KGKB03] considers a database of n = 100000 points and uses formula (1) to examine the sensitivity of s to various parameters, and gives the results in Figure 3.5. Figure 3.5 presents the minimum sample size required to guarantee that a fraction of ϕ=0.2 of points in $|u|$ is included in the sample, for a range of probabilities, for $|u|$ ranging from 100 to 100000 points.

It is evident from Figure 3.5 that, using uniform random sampling, it cannot be guaranteed with high probability that a large fraction of a small cluster is included in the sample, unless a very large sample is used. This requirement, however, defies the motivation of sampling.

Thus for applications where small clusters are important patterns to be detected and analyzed, uniform random sampling is not a proper technique to be used. Biased sampling [KGKB03], which is able to include small clusters in a smaller sample, is a better choice for such applications. However, for applications in which small clusters are unimportant, uniform random sampling is still a proper choice.

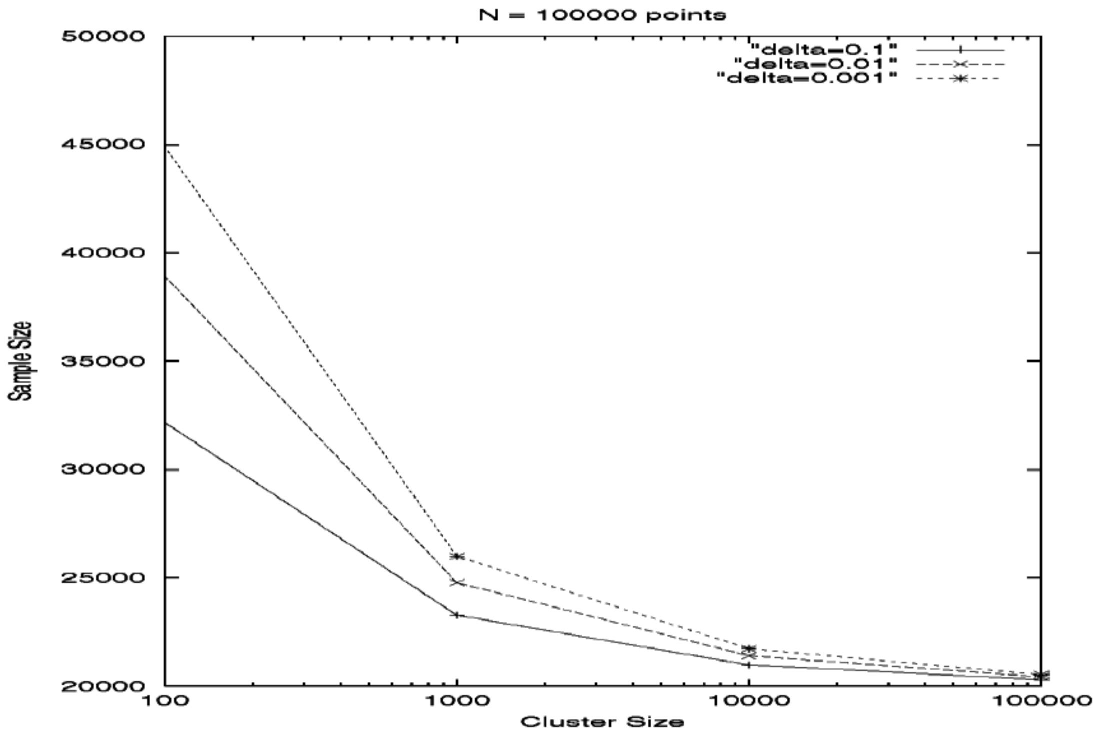


Figure 3.5: The sample size required by uniform random sampling to guarantee that a cluster of given size is included in the sample with probability no less than delta [KGKB03]. This figure is used without the authors' permission.

It also needs to be noted that under certain scenarios a non-trivial sampling method might produce significant information loss with a small change in the sampling rate. Thus it is important to suggest proper sampling rate change granularity to users if they are allowed to change the sampling rate. The following example shows how page-level bernoulli sampling might produce significant information loss with a small change in the sampling rate in the situation that the number of rows contained in each page is large. Similar analysis can be found in [HK04], but I present the detailed proof while [HK04] does not.

Consider the following query:

SELECT SUM(t.a1) / q

FROM table1 TABLESAMPLE page-level-sampling (sampling-rate q)

The information loss is measured using standard error of the estimator [HK04]. According to formula (4) in [HK04], the standard error of this query using a pure page-level Bernoulli sampling is:

$\sqrt{(1/q-1)\sum_{j\in U}\alpha_j^2}$, where q is the sampling rate, U is the set of pages in the table, and α_j is the sum of $a1$ of all tuples on page j of the table.

Assume that a1 = M in all tuples in the table, and there are N tuples in the table.

If each page contains one row (in this case the sampling is the same as a row-level Bernoulli sampling), the standard error of the estimator is $\sqrt{N} * \sqrt{(1/q-1)} * M$.

If each page contains K rows, the standard error of the estimator is $\sqrt{K} * \sqrt{N} * \sqrt{(1/q-1)} * M$.

It can be seen that page-level sampling magnifies the standard error of row-level sampling by a factor of $\sqrt{K}$. Thus if K is large, the information loss caused by a small change in the sampling rate might be large.

From the above examples, it can be seen that it is important to measure the information loss caused by a sampling approach. In the following sections, I give some examples of measures of information loss found in the sampling literature.

Standard Error of Unbiased Estimator

In [HK04], bi-level Bernoulli sampling is used to estimate answers of aggregation queries. The information loss is measured using the standard error of the unbiased estimator, which is defined as the square root of the variance of the estimator. The larger the standard error the estimator has, the larger amount of information loss could be introduced by the sampling approach. Theorem 1 in [HK04] gives formulae for calculating the variance of the estimators of bi-level Bernoulli sampling for SUM, COUNT and AVG queries, and presents how to estimate the variance of the estimators using a sample.

Group Loss and Average Relative Error

In [BCD03] and [AGP00], sampling is used to estimate answers of aggregation queries with group-bys. Two accuracy criteria are used to measure the information loss. The first criterion is that as many of the groups as possible should be preserved in the approximate answer. It is measured by the percentage of groups missed by the approximate answer. The second criterion is that the error in the aggregate value for each group should be small. It is measured by the average relative error on the exact answer of the approximate answer. For a group missed by the approximate answer, the relative error is estimated as 100%. For analytical convenience, the average squared relative error on the exact answer of the approximate answer is used because it is more analytically tractable than the average relative error. The expressions of the above criteria, and the equations for calculating expected values of the average squared relative errors for both uniform sampling and small group sampling are presented in [BCD03].

Distinguishable Visualizations

As suggested by [DE02], information loss caused by sampling in visualization could be measured by estimating how the visualization obtained from the sample is distinguishable from that obtained from the full data set. For example, if the error of the height of a vertical histogram bar caused by sampling is less than one pixel, the information loss is small enough and the sample size is big enough.

Visual-Based Comparison

The displays generated with and without sampling can be visually compared by human beings. For example, [WFA$^+$03] put the scatterplot displays generated with and without sampling together (Figure 3.6) to illustrate that the major patterns were preserved in the scatterplot displays after sampling is applied.

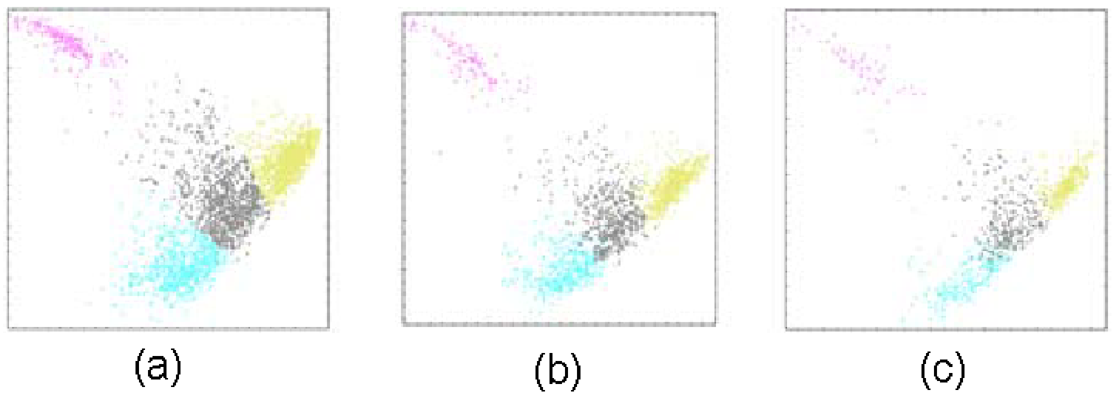


Figure 3.6: The scatterplot displays generated by (a) all original 3298 data items, (b) 1649 samples, and (c) 824 samples [WFA+03]. It appears they contain similar patterns. This figure is used without the authors' permission.

Although visual-based comparison is fast and convincing, it needs human attention and interpretation and thus is not practical in some applications. Besides, visual-based comparison is not always reliable. [WFA+03] gave an example where two scatterplots look fairly different but actually contain very similar patterns (Figure 3.7).

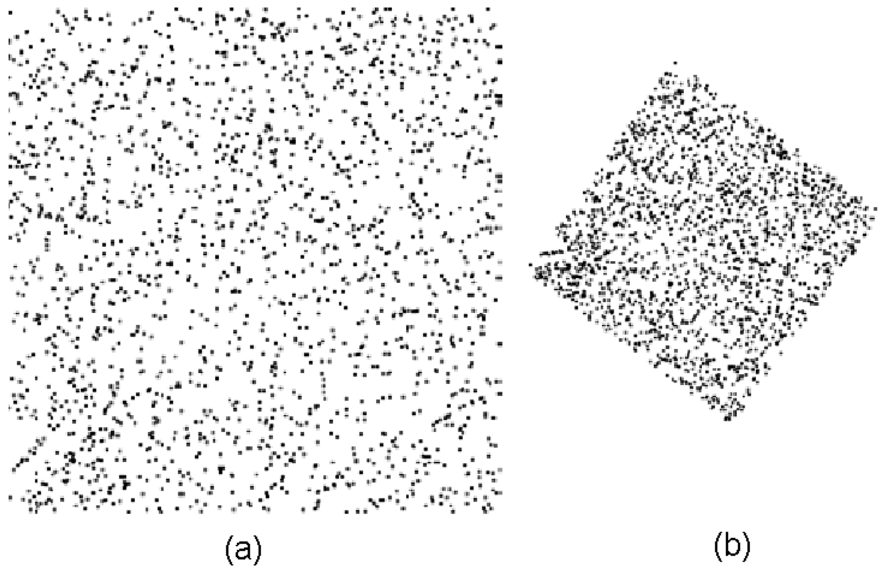


Figure 3.7: These two scatterplot displays contain very similar patterns but our eyes are fooled by the lack of visible patterns that are required to register the images [WFA+03]. This figure is used without the authors' permission.

Statistical Analysis of Generated Visualizations

Rather than analyzing the information loss of the direct result of sampling, information loss can be measured by analyzing the final images generated with and without sampling. I call the former one as *direct analysis* and the latter one as *indirect analysis*. Examples of direct analysis are calculating standard error of unbiased estimator of the sampled data and calculating group loss and average relative error of the sampling results.

[WFA$^+$03] gives an example of indirect analysis that uses procrustes analysis [CC94] to measure the difference between the scatterplot displays generated with and without sampling to measure information loss caused by sampling in visualization.

It is interesting that indirect analysis might be more efficient than direct analysis since the number of pixels in a display is often less than the number of visual entities displayed, and non-distinguishable differences caused by sampling are automatically suppressed at the image level.

All simplification operators cause information loss. In the following discussion of other simplification operators, information loss will no longer be discussed separately since similar to the sampling operators, information loss of those operators can also be measured using statistical methods and visual approaches.

View Continuity

In an interactive visualization with sampling, when the sampling rate changes, maintaining continuity in sampling, both within a session and possibly between sessions, is recommended [DE02]. The reason is that smooth transitions between different resolutions are beneficial to the user [Spe01]. In the Astral Telescope Visualiser [DE02], the authors suggested that the following strategies can be used to maintaining continuity among different views:

- When change from a view with a higher sampling rate to a view with a lower sampling rate, sample from the data displayed in the view of higher sampling rate.
- Record the samples used in views of different sampling rates. Redisplay the recorded sample if a view of the same sampling rate is requested again.

However, sometimes resampling is intentionally used in order to remind the users that sampling is being used for abstraction.

3.2 Aggregation Operators

Definition: *Aggregation* refers to a simplification in which a relationship between objects is regarded as a higher level object [SS77]. In making such a simplification, many details of the relationship may be ignored. Aggregation is an approach whose input is an object set $A = (a_1, a_2, ..., a_i)$ and whose output is another object set $B = (b_1, b_2, ..., b_j)$, $0 < j < i$. Each object in B is generated by a subset of objects in A. Objects in B have a different attribute set from objects in A.

There are many aggregation operators, such as clustering, histograms, wavelet transformation, and data cubes. I discuss some of them in more detail, and similar discussions can be applied to other aggregation operators.

3.2.1 Clustering Operators

Definition: *Clustering* is a division of data into groups of similar objects. Each group, called a cluster, consists of objects that are similar among themselves and dissimilar to objects of other groups [Ber02].

Clustering falls into the aggregation category since a cluster is regarded as a higher level object that represents all objects composing it and hides details of the relationships among those objects.

Clustering is an important simplification operator often used in multi-resolution visualization systems. It is widely used because of two reasons:

- By visualizing clusters rather than the original data the number of visual elements displayed can be greatly reduced.
- Clustering itself is a pattern discovering process. Thus visualizing clusters explicitly reveals hidden patterns to viewers.

Clustering has been studied in considerable detail by the statistics, pattern recognition, machine learning, and database management fields for different domains of data [Ber02]. [Ber02] gives a thorough survey of various clustering algorithms. In the following sections, I briefly introduce some popular clustering approaches. I use the categorization of clustering algorithms proposed in [Ber02], and add a category of human-computer clustering approach to it. The definitions of most clustering approaches come from [Ber02], as well as critiques to these approaches.

Hierarchical Clustering Algorithms

Hierarchical clustering builds a cluster hierarchy, or in other words, a tree of clusters, also know as a dendrogram. Every cluster node contains child clusters; sibling clusters partition the objects covered by their common parent [Ber02]. Such an approach allows exploring data on different levels of granularity.

Hierarchical clustering methods are categorized into agglomerative (bottom-up) and divisive (top-down) approaches [JD88, KR90]. An agglomerative clustering starts with one-object clusters and recursively merges two or more most appropriate clusters. A divisive clustering starts with one cluster of all objects and recursively splits the most appropriate cluster. The process continues until a stopping criterion (frequently, the requested number k of clusters) is achieved.

In hierarchical clustering methods, the distances or similarities between objects are used rather than the attributes of the objects. Hierarchical clustering methods can be further divided into multiple categories according to how the distance between subsets of objects is decided when merging or splitting subsets of objects, which is called the *linkage metrics*. *Graph methods* use the minimum, maximum, or average of the distances measured for all pairs of objects with one object in the first set and another object in the second set as the linkage metrics. Examples of graph methods includes SLINK [Sib73] and CLINK [Def77]. *Geometric methods* use the central point of a cluster to represent it. Traditional linkage metrics-based hierarchical clustering suffers from time complexity. Many new hierarchical clustering algorithms have been proposed in order to reduce time complexity.

For example, CURE [GRS98] is a hierarchical agglomerative clustering algorithm targeting at multi-dimensional data sets with a large number of data items and a low number of numerical attributes. A major feature of CURE is that it represents a cluster by a certain fixed number of points that are generated by selecting well-scattered points from the cluster and then shrinking them toward the center of the cluster by a specified fraction. The distance between two clusters used in the agglomerative process is equal to the minimum of distances between two scattered representatives. Thus CURE takes a middle-ground approach between the graph (all-points) methods and the geometric (one centroid) methods. CURE achieves scalability by employing a combination of random sampling and partitioning. A random sample drawn from the data set is first partitioned and each partition is partially clustered. The partial clusters are then clustered in a second pass to yield the desired clusters.

Partitioning Clustering Algorithms

Partitioning algorithms divide data into several subsets. Since examining all possible subset systems is computationally infeasible, some partitioning methods iteratively reassign points between the clusters until an objective function is optimized. These methods are called *relocation algorithms*. Some relocation algorithms use objective functions depending on probabilistic models and are named *probabilistic clustering*. Some relocation algorithms use objective functions depending on a partition and can be subdivided into k-medoid and k-mean methods according to how representatives of clusters are constructed.

In the probabilistic approach, data is considered to be a sample independently drawn from a mixture model of several probability distributions [MB88]. The area around the mean of each distribution constitutes a natural cluster. The cluster can be associated with the corresponding distribution's parameters such as mean and variance. The goal of the clustering algorithms is to maximize the overall probability or likelihood of the data, given the final clusters. The EM (Expectation-Maximization) method [MB88] is a classic probabilistic clustering approach, and many other methods have been developed to accelerate and improve it, such as FREM [OO02]. The EM methods accommodate categorical variables as well as numeric variables.

In k-medoid methods a cluster is represented by one of its points, which is called a *medoid*. When medoids are selected, clusters are defined as subsets of points close to respective medoids, and the objective function is defined as the averaged distance or another dissimilarity measure between a point and its medoid. The k-medoid methods resist outliers well since peripheral cluster points do not affect the medoids. Among the k-medoid methods, CLARANS [NH94] scales well with data sets with a large number of points. CLARANS considers a graph whose nodes are the sets of k medoids and an edge connects two nodes if they differ by exactly one medoid. It uses random search that starts with an arbitrary node and randomly checks some maximum number of neighbors. If a neighbor represents a better partition, the process continues with this new node.

Different from k-medoid methods, k-mean methods represent a cluster by the mean (weighted average) of its points, the so called *centroid*. This approach does not work well with categorical attributes, but it works well with numerical attributes. The sum of discrepancies between the points and their centroids expressed through appropriate distance is used as the objective function. Forgy's algorithm [For65] is a typical k-mean approach. It iteratively reassigns all the points to their nearest centroids, and recomputes centroids of newly assembled groups until a stopping criterion (such as no reassignments happen) is achieved. Different from Forgy's algorithm, another group of k-mean approaches iteratively reassign points based on more detailed analysis of effects on the objective function caused by moving a point from its current cluster to a potential new one. K-mean methods have been widely studied and are by far the most popular clustering tool used.

Density-based partitioning methods are another family of partitioning clustering algorithms. In these approaches, a cluster is defined as a connected dense component growing in any direction the density leads. Therefore, density-based algorithms are capable of discovering clusters of arbitrary shapes that are not rectangular or spherical. There are two major types of density-based methods. In the density connectivity approaches, all points reachable from core objects (a core object is an object with a neighborhood consisting of more objects than a given threshold) can be included into maximal connected components serving as clusters. DBSCAN [KSSB90], OPTICS [ABKS99], and DBCLASD [XEKS98] are examples of density connectivity methods. The density function approaches compute density functions defined over the underlying attribute space to detect clusters. DENCLUE [HK98] is an example of the density function methods.

Grid-based Clustering Algorithms

In *grid-based clustering* methods, data partitioning is induced by points' membership in segments (a segment is a direct Cartesian product of individual attribute sub-ranges) resulting from space partitioning, while space partitioning is based on grid-characteristics accumulated from input data. Accumulation of grid-data makes grid-based clustering techniques independent of data ordering. Grid-based methods work with attributes of different types. CLIQUE [AGGR98] is an important grid-based clustering algorithm that also combines the idea of density-based clustering and hierarchical clustering. It scales to data sets with large numbers of dimensions and data items, and is able to detect clusters embedded in subspaces of high dimensional data without requiring the users to guess subspaces that might have interesting features.

Other Clustering Algorithms

[Ber02] also surveys other clustering algorithms, such as co-occurrence clustering of categorical data, constraint-based clustering, methods making use of artificial neural networks, and co-clustering methods for large scale data sets. I refrain from describing these methods, otherwise the clustering operator will occupy too many pages in the dissertation. However, there is a group of clustering methods that have not received special attention in [Ber02]. I introduce them in the following sections since they are closely related to visualization.

Human-Computer Clustering Approaches

Human-computer clustering approaches refer to the clustering methods that are a combination of human intuition and observation and computational support by the computer. It is claimed that the meaningfulness and definition of a cluster are best characterized with the use of human intuition [Agg01]. Computational support is necessary since clustering is a complex task. Many human-computer clustering approaches arise, such as the IPCLUS clustering approach described in [Agg01].

In the IPCLUS clustering approach [Agg01], user interactions are provided to allow users to visually separate the clusters in different data projections. In this approach, the density profile of a data projection is visually presented (Figure 3.8). Clusters correspond to dense regions in the data and are represented by peaks in the density profile. The regions that separate out the different clusters have low density values and are represented by valleys in the density profile. In order to actually isolate the clusters, the user can interactively specify density value thresholds that correspond to the noise level at which clusters can be separated from one anther. Figure 3.8 (a) and (b) show that different numbers of clusters are formed according to different user defined density value thresholds.

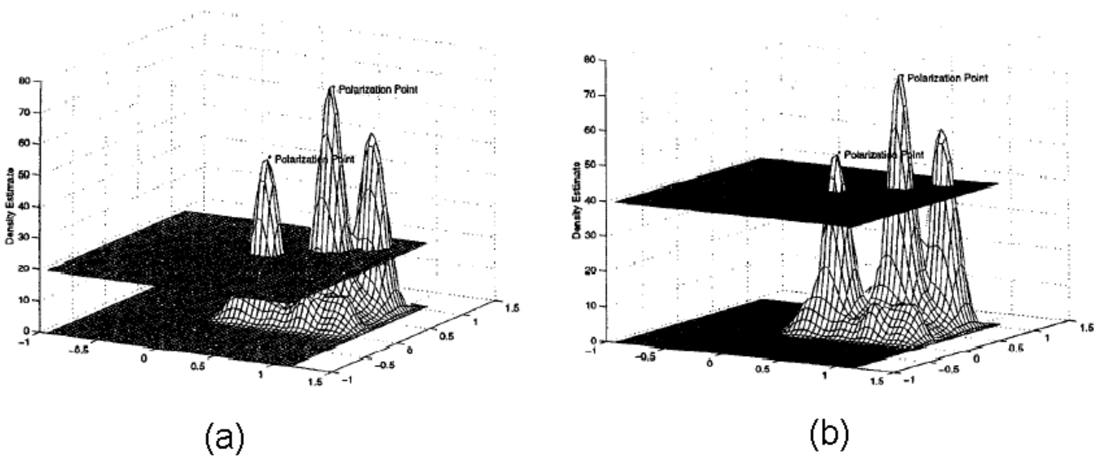


Figure 3.8: The IPCLUS algorithm allows users to interactively select density value thresholds to separate clusters [Agg01]. (a) Two clusters are formed with a lower density value threshold. (b) Three clusters are formed for the same data projection with a higher density value threshold. This figure is used without the authors' permission.

Visual Representation of Hierarchical Clustering Results

A natural way to convey the results of hierarchical clustering is to visualize the generated hierarchy using hierarchy visualization techniques. Hierarchies generated from clustering algorithms are often very large and challenge existing hierarchy visualization techniques.

Most recent hierarchy visualization research has concentrated on the challenge of displaying large hierarchies in a comprehensible form [DE01]. Among the efforts, there are improved node and link diagrams [Fur86, BKW89], cone trees [CRM91, RMC91] and their variations [KY93, CK95, vHvdWvW01], collapsible cylindrical trees [DE01], botanical trees [KvdWvW01], radial layouts [BETT99, Ead92, YFDH01], hyperbolic layouts [LRP95, LR96, MB95, Mun97, Mun98], skeletal images [HMM$^+$99], and castles [KH81].

Space-filling hierarchy visualization techniques are a significant category among all the techniques in that they use display space very efficiently. Among the space-filling approaches, there are rectangular space-filling techniques, such as the treemap and its variations [JS91, Shn92, WS99, SW01], and radial (also called circular) space-filling techniques [AH98, Chu98, SZ00, YWR02].

Visual Representation of Clusters

When few clusters are used to describe a large number of objects, cluster properties such as densities and shapes need to be represented. Cluster representation is an important issue when the clustering operator is used and many examples can be found from existing multi-resolution visualization systems.

For example, Hierarchical Parallel Coordinates [FWR99b] use a mean and a band to convey the mean and extents of a cluster, and opacity to convey the cluster population in a multi-dimensional space, which is shown in Figure 3.9. The mean-band method will be described in more detail in Chapter 4. The InfoSky visual explorer [GKS$^+$04] visualizes document clusters using polygons to allow easy selection of the clusters, which is shown in Figure 3.10. The SpaceTree [PGB02] uses triangular icons to visually represent clusters (see Figure 3.11). Darker icons correspond to clusters with more nodes in the branches. Taller icons correspond to clusters with deeper branches, while wider icons correspond to higher average branching factors (number of nodes divided by the depth). [KLS00] uses a similar approach to represent clusters, which is shown in Figure 3.12.

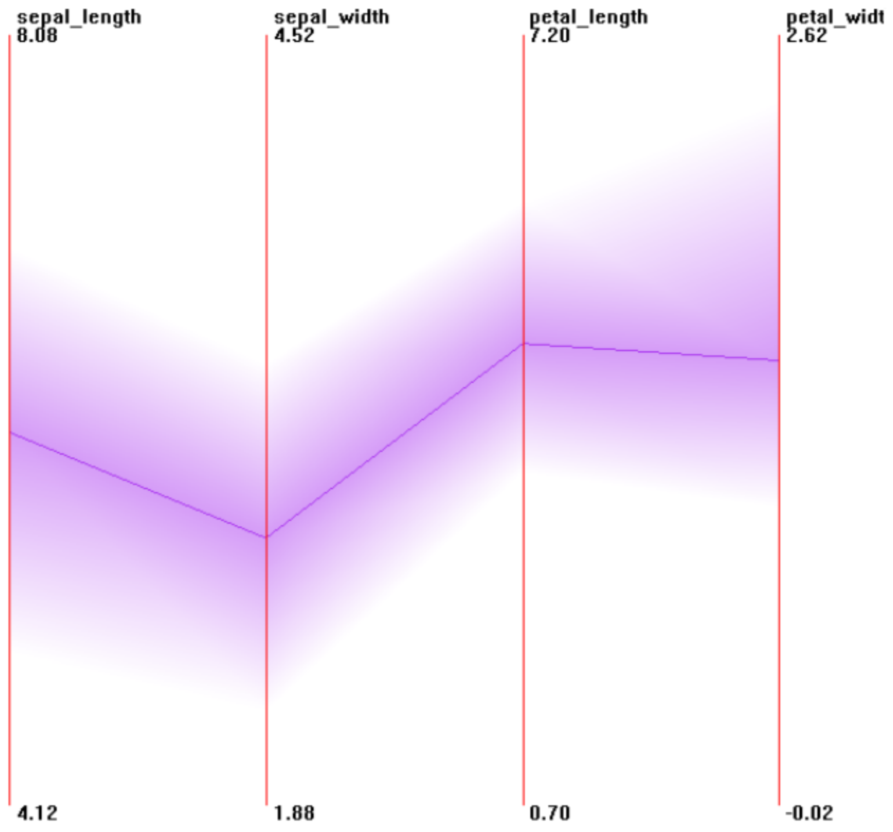


Figure 3.9: A cluster in the Iris data set visualized using a mean and a band. The band indicates the extents of the cluster.

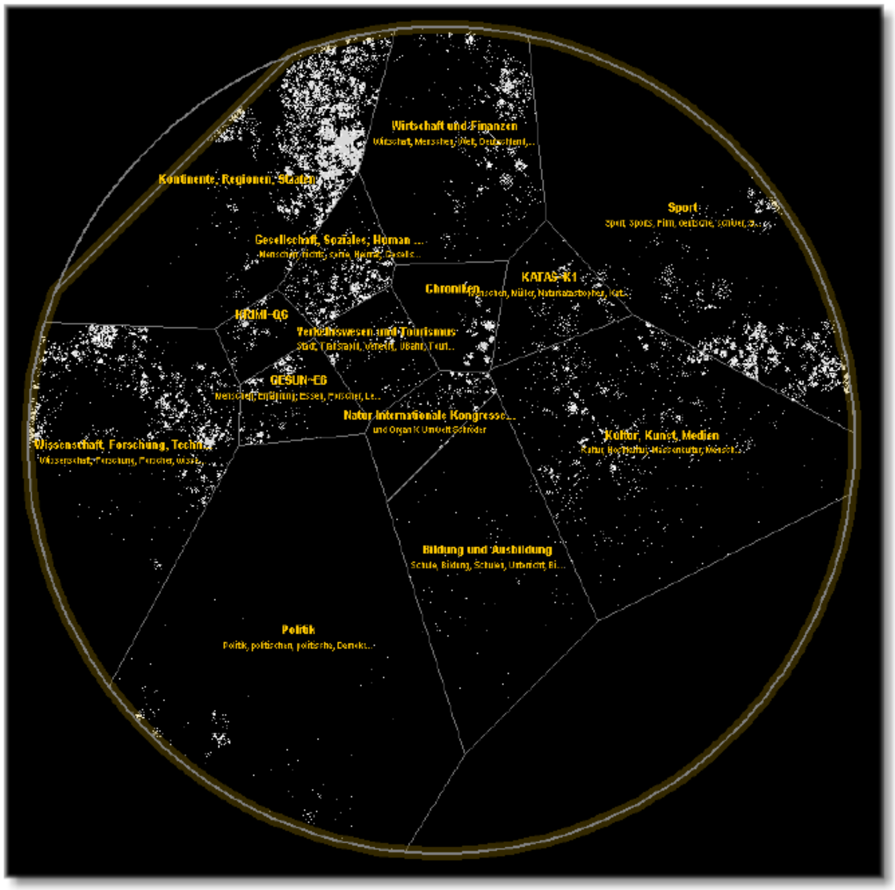

Wirtschaft und Finanzen
Kontinente, Regionen, Staaten
Sport
Gesellschaft, Soziales, Human ...
KATAS-K1
Chroniken
KRIMI-QG
Verkehrswesen und Tourismus
GESUN-EG
Natur Internationale Kongresse...
Kultur, Kunst, Medien
Wissenschaft, Forschung, Techn...
Bildung und Ausbildung
Politik

Figure 3.10: The InfoSky visual explorer [GKS+04] visualizes document clusters using polygons to allow easy selection of the clusters. This figure is used without the authors' permission.

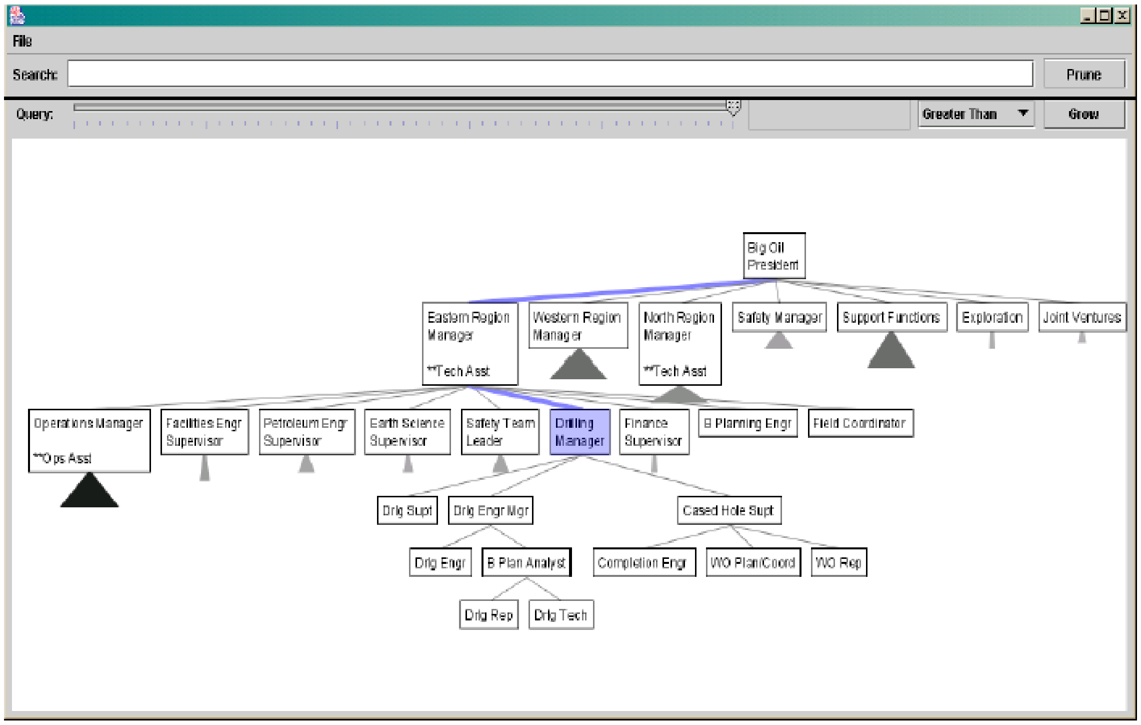


Figure 3.11: The SpaceTree [PGB02] uses triangular icons to visually represent clusters in the tree. Darker icons correspond to clusters with more nodes in the branches. Taller icons correspond to clusters with deeper branches. While wider icons correspond to higher average branching factors (number of nodes divided by the depth). This figure is used without the authors' permission.

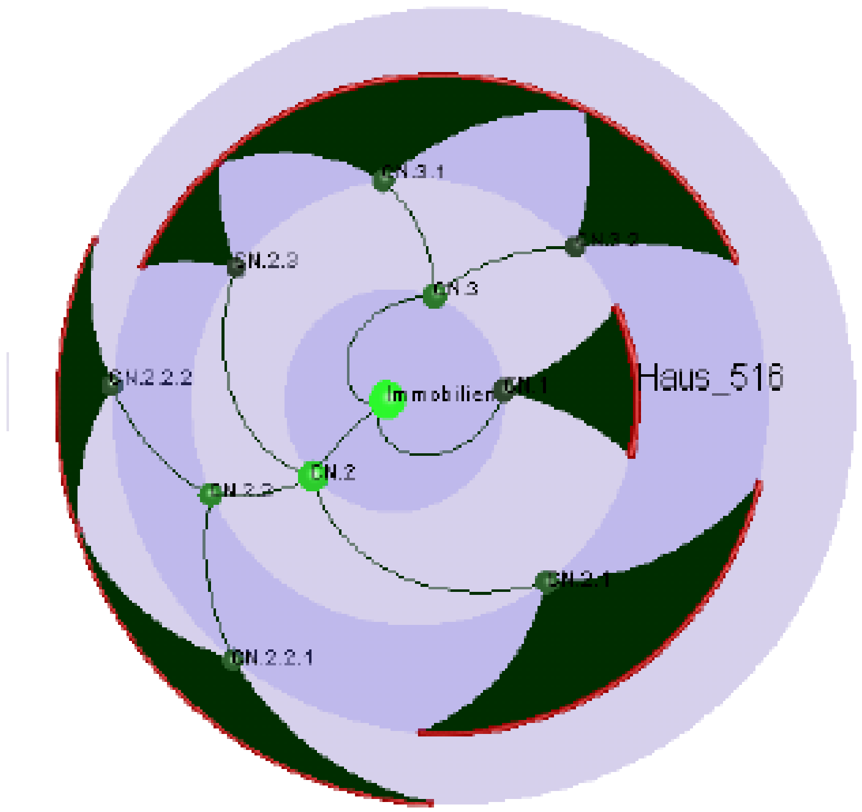


Figure 3.12: Icons are used to visually represent clusters in the tree projected onto a hemisphere [KLS00]. This figure is used without the authors' permission.

3.2.2 Histogram Operators

Definition: A *histogram* partitions the data space into buckets. In each bucket, the data distribution is often assumed uniform and recorded using simple statistic data. The distribution of each bucket can also be approximated using more complex functions and statistical data.

Histograms fall into the aggregation category since a bucket is regarded as a higher-level object that represents all objects falling into it and hides details of the relationships among those objects.

Histograms are used to capture important information about the data in a concise representation [WS03]. In the following sections, I discuss some popular histogram techniques.

One-Dimensional Histogram

One-dimensional histograms provide aggregated information for data in one-dimensional space. Equi-width histogram is the simplest histogram. It is obtained by splitting the range of the data into equal-sized bins. Then for each bin, the number of data items in the data set falling into it is counted. A one-dimensional histogram can be visually represented by a bar chart whose vertical axis is the counts for each bin, and whose horizontal axis is the response variable [NIS05]. Figure 3.13 shows several Equi-width histograms of a SAT math scores data set with various bin sizes (the figures were captured from [NIS05]). The histograms shown in Figure 3.13 (a) and (b) approximate the data distribution well while the histograms shown in Figure 3.13 (c) and (d) have big information loss.

In order to improve accuracy and efficiency, other one-dimensional histogram techniques, such as Equi-depth [PSC84], End-biased [IP95], Max-diff [PHIS96], V-optimal [JKM$^+$98], and Spline-optimal [KW99] histograms have been proposed. They allow un-

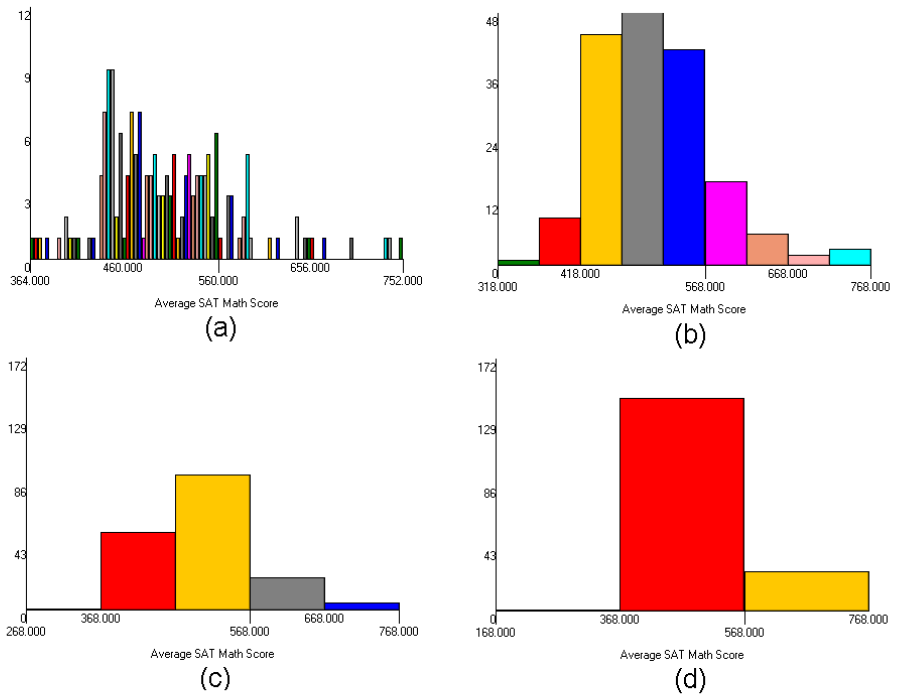


Figure 3.13: Equi-width histograms of a data set that containing the SAT math scores of students in some colleges with various bin sizes. These figures were captured from [NIS05].

equal sized bins or/and use more complex functions and statistic data to decrease the number of bins used and improve the accuracy of the data distribution representation.

One-dimensional histograms can be used to represent the data distribution in a multi-dimensional space based on the assumption that each dimension is independent from the others in the space. Upon this assumption, the data set is projected to each of the dimensions composing the multi-dimensional space, and a one-dimension histogram is constructed upon each projection. For a given multi-dimensional range (which is often called a *query* in the multi-dimensional space), the number of data items falling into this range (which is often called the *selectivity* of the query) is estimated in this way: project the query on each dimension, estimate the selectivity of each one-dimensional query using the one-dimensional histograms, and multiply the selectivities from all one-dimensional queries to get the selectivity of the multi-dimensional query.

One-dimensional histograms have been widely used in multi-resolution visualization systems. For example, [DHMR99] uses one-dimensional histograms to provide feedback for dynamic queries. In Figure 3.14 (a), two one-dimensional histograms are shown. One shows the birth date distribution of people in a data set whose birth dates are between 8/1935 and 8/1968. The other one shows the salary distribution of people in the same data set whose salaries are between $15,760 and $66,729. Using dynamic query, 72 people in the data set have been selected. The blue bars in the histograms show the distribution of the attribute values for the 72 selected people. The selected data are shown in Figure 3.14 (b) as a scatterplot.

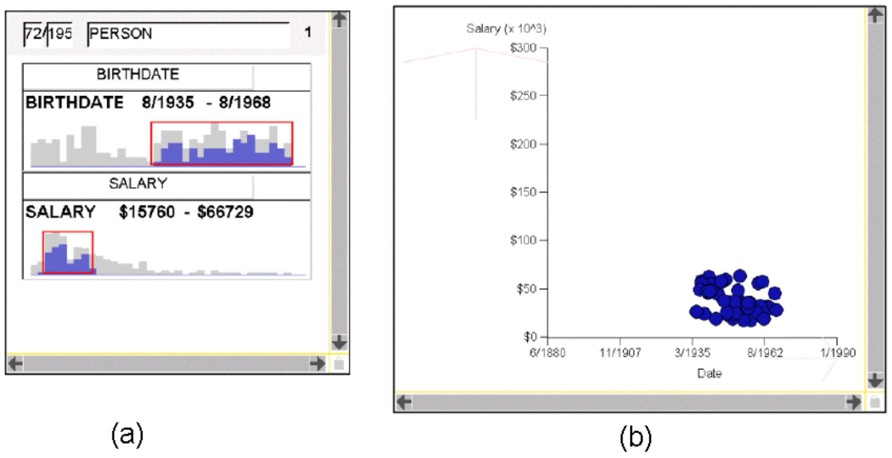


Figure 3.14: Use one-dimensional histograms to provide feedback for dynamic query [DHMR99]. This figure is used without the authors' permission.

For another example, ThemeRiver [HHN00] uses a river-like visualization to represent multiple 1D histograms upon the same time period in the same display. ThemeRiver is a prototype that visualizes thematic variations over time within a large collection of documents. The strength of each theme over time forms a one-dimensional histogram. There are many themes so that there are many one-dimensional histograms. In order to reveal the thematic variations over time, these histograms need to be presented in the same display. [HHN00] explores two approaches. The first one uses one bar to represent a time slice, and the color variations and sizes within the bar represent the relative strength of themes specific to that slice (see Figure 3.15). This approach generates a poor display since it requires users to work hard at integrating a theme across time because its baseline may vary considerably in the bars. Thus [HHN00] propose a second approach, the river-like display (see Figure 3.16). The "river" flows from left to right through time, changing width to depict changes in the thematic strength of temporally associated documents. Colored "currents" flowing within the river narrow or widen to indicate decreases or increases in the strength of an individual topic or a group of topics in the associated documents.

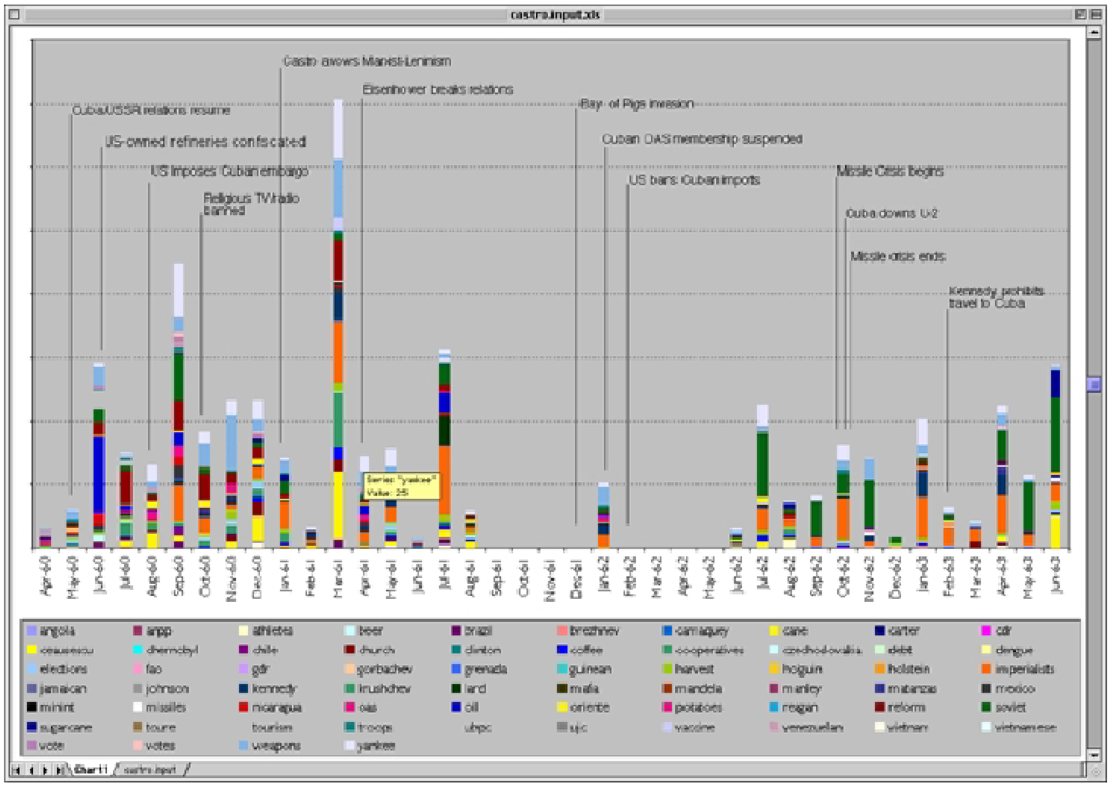


Figure 3.15: This histogram depicts changes in thematic content over time for the Castro collection [HHN00]. This figure is used without the authors' permission.

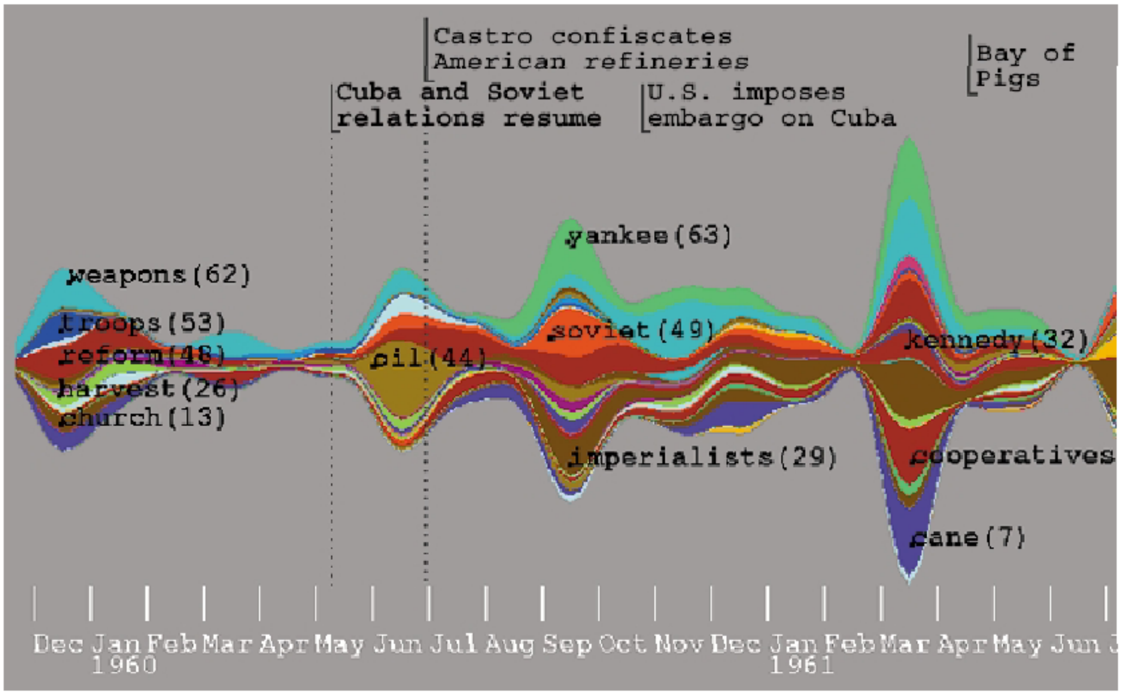


Figure 3.16: The themeriver display that depicts changes in thematic content over time for the Castro collection [HHN00]. This figure is used without the authors' permission.

Multi-dimensional Histograms

Using one-dimensional histograms to represent the data distribution in a multi-dimensional space is based on the assumption that each dimension is independent from the others. This assumption is often false in a multi-dimensional space. In order to improve the accuracy of data distribution representation in multi-dimensional spaces, *multi-dimensional histograms* that divide a multi-dimensional space into buckets have been proposed.

Traditional multi-dimensional histograms divide the multi-dimensional space into non-overlapped buckets. For example, in the PHASED histogram [MD88], a dimension is first chosen and the multi-dimensional array is split along that dimension into several buckets in which the sums of measure values are nearly equal. Each bucket is then treated as a (d-1)-dimensional array and is split recursively for the remaining dimensions.

The histograms that do not allow buckets to overlap suffer from a conflict when they are applied on data sets with real attributes: smaller buckets can capture the variations in the data distribution more accurately than larger buckets. The number of non-overlapping buckets increases greatly with the decrease of the bucket sizes for multi-dimensional data sets. However, on the other hand, with more buckets, the number of buckets that a query can partially intersect increases rapidly for multi-dimensional data sets, which is a factor decreasing the overall quality of the selectivity estimation.

To solve the above conflict, histogram techniques such as STHoles [BCG03] that allow buckets to overlap and to be of variable sizes have been proposed. The benefit is that more regions of different densities can be expressed using the same number of rectangular buckets if buckets are allowed to overlap and be of variable sizes. The STHoles [BCG03] histogram is constructed by iteratively "smoothing" data density in the multi-dimensional space. In each iteration, the space is partitioned using a regular grid. Dense buckets with the largest average densities are identified. Rather than removing all data points from a dense bucket, only some data points are removed from it so that the density of the bucket becomes approximately equal to its surrounding areas. A bucket of the density of the removed data points is recorded. In this way, the density of the entire data set is "smoothed" out. Thus in the next iteration, a coarser grid can be used. In an overlapped area, the density is the sum of the overlapping buckets (a data point will only be counted in one of the overlapping buckets).

Figure 3.17 shows different histograms built for a multigaussian data distribution [BCG03]. All histograms use the same amount of memory. It can be seen that STHoles histogram captures the data distribution better than all other histograms.

All the histograms discussed thus far are static histograms. Multiple passes of the data set are usually required in order to build a static histogram. It is expensive for a large data set. In addition, it is expensive to adapt static histograms to underlying data distribution changes. "Dynamic" histograms, such as STGrid [AC99] and STHoles [BCG03], have

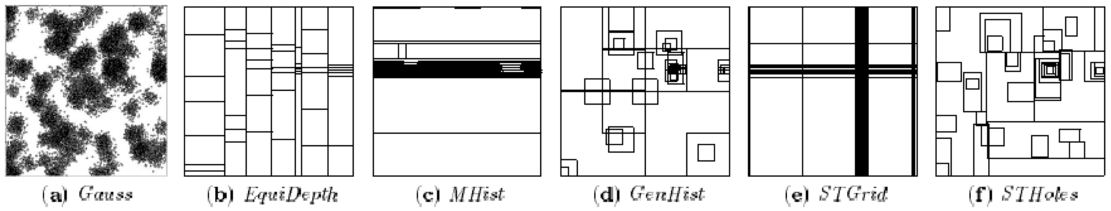


Figure 3.17: The Guass data set and its histograms [BCG03]. (a) The Guass data set (b) EquiDepth histogram (c) Mhist histogram (d) GenHist histogram (e) STGrid histogram (f) STHoles histogram. This figure is used without the authors' permission.

been proposed to solve these problems. There are two key ideas in the self-tuning histogram techniques. First, query results reveal the data distribution of the queried data set. Thus a histogram can be constructed by making use of query results, which are "free", without accessing the data. Second, the workload reveals heavily queried areas and can be used to allocate more resource to these areas. The lifecycle of a self-tuning histogram consists of two stages. First, a coarse histogram is built without accessing the data. Then, the coarse histogram is refined using query feedbacks. In particular, by comparing the frequencies of buckets in histogram involved in the query range with the actual query results, inaccurate buckets (such as those that have non-uniform densities or have inaccurate frequencies) can be identified and adjusted. Buckets have similar frequencies can be merged in order to reduce the number of the buckets in the histogram.

Dimensional stacking [WLT94] is one of the multi-resolution information visualization techniques that make use of multi-dimensional histograms. It is a N-D histogram and uses cell colors to indicate bin populations. I believe that it will be a trend to develop more multi-resolution visualization systems using multi-dimensional histograms. For example, an approximate display of a multi-dimensional data set could be generated without accessing the data set by visually representing the boundaries and densities of buckets in a multi-dimensional histogram. For another example, in a visual exploration focused on finding outliers in a data set, if there is a multi-dimensional histogram available, areas covered by dense buckets in the histogram could be filtered out, thus only data items from areas covered by sparse buckets need to be fetched and visualized.

3.2.3 Wavelet Operators

Definition: *wavelets* - filter matrices that accept a data stream with items, and generate items of approximations and items of details. The items of approximation is a coarse summary of the original data, and items of details contain the data loss during the decomposition [WB96].

Figure 3.18 [WB96] depicts the process of a wavelet decomposition. Each of the four neighboring points of the finer data (upper curve) is used to compute the corresponding approximation of the coarser data (lower curve). The calculation involves precomputed matrices which are described in [Dau92]. The exact number of data points used in the downsizing process is related to the number of vanishing moments of an orthogonal wavelet. A hierarchy of coarse approximations is generated when this process is applied iteratively to the approximations to obtain increasingly coarse data [WB96].

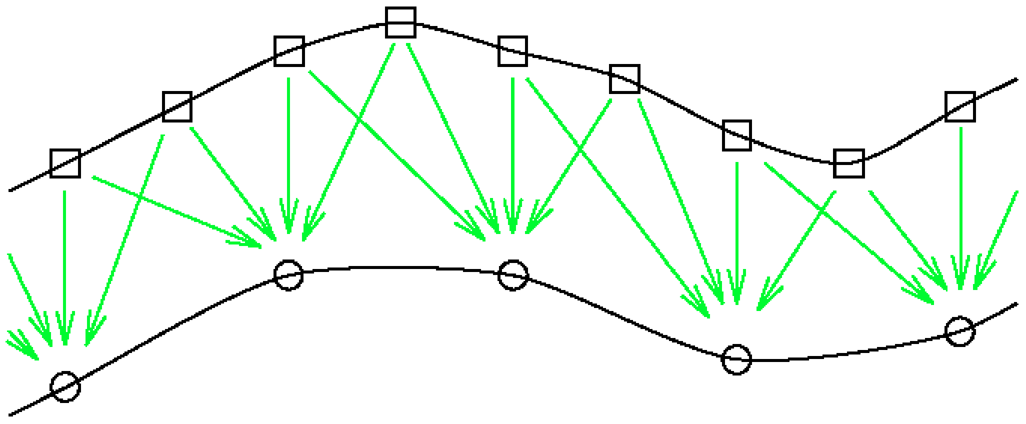


Figure 3.18: A fine data curve is downsized to a coarse one using wavelet decomposition [WB96]. This figure is used without the authors' permission.

Wavelets are widely used in multi-resolution visualization systems. For example,

[WB96] proposed a multi-resolution multi-dimensional brushing using wavelet transformation. Figure 3.19 [WB96] shows a simplified example of the brush on a bi-variant scatterplot. It can be seen that the coarse background outside of the brush gives the overall structure of the context surrounding the brushed area while the fine brushed data provides the local structure inside the brushed area.

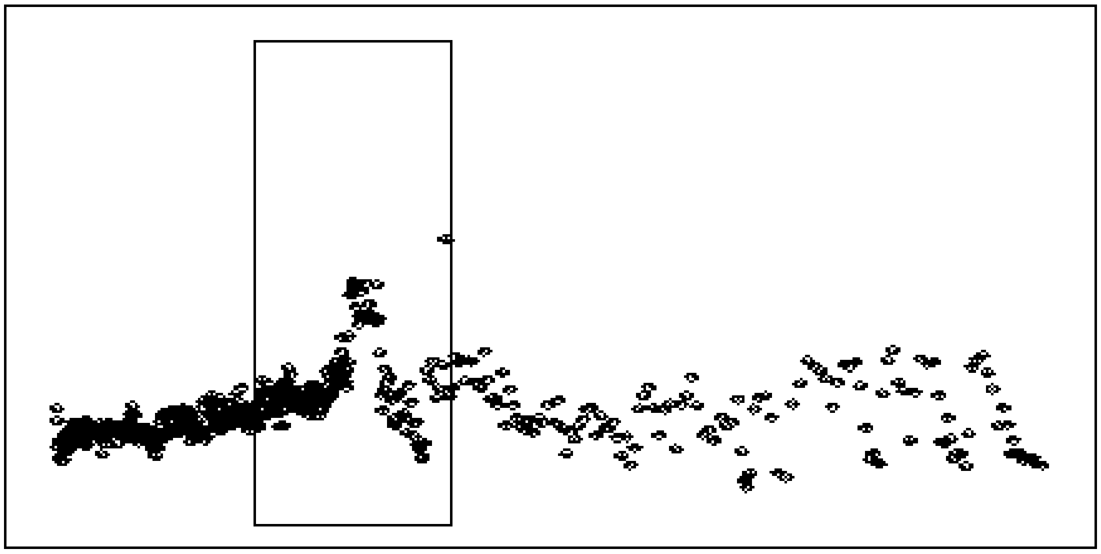
3.2.4 Other Aggregation Operators

Besides the clustering, histogram, and wavelet operators, there are many other simplification operators, such as data cube, that belong to the aggregation operator category.

Data cube is a relational aggregation operator generalizing group-by, cross-tab, and sub-totals [GCB$^+$97]. It is a mechanism for performing data abstraction that provides summaries of the underlying data at different meaningful levels of detail [SH00, STH02]. Figure 3.20 shows how a three-dimensional data cube is projected [SH00]. Data cube is used by multi-dimensional visualization systems, such as the one presented in [STH02], to provide meaningful aggregations of the data items to users.

For another example, a long document is often aggregated into one or a few sentences that summarize the key features of the document in document visualization. At a low level of detail display, these sentences, rather than the whole document, are visualized. This is also an aggregation operator. Figure 3.21 shows a search result displayed by www.google.com. Each web page in the result set is visually represented by one sentence describing how the web page is related to the search keywords.

It is almost impossible to list all aggregation operators. However, they share the feature of hiding the relationships among a group of objects into a high-level object. Aggregation operators are powerful simplification tools.


(a)

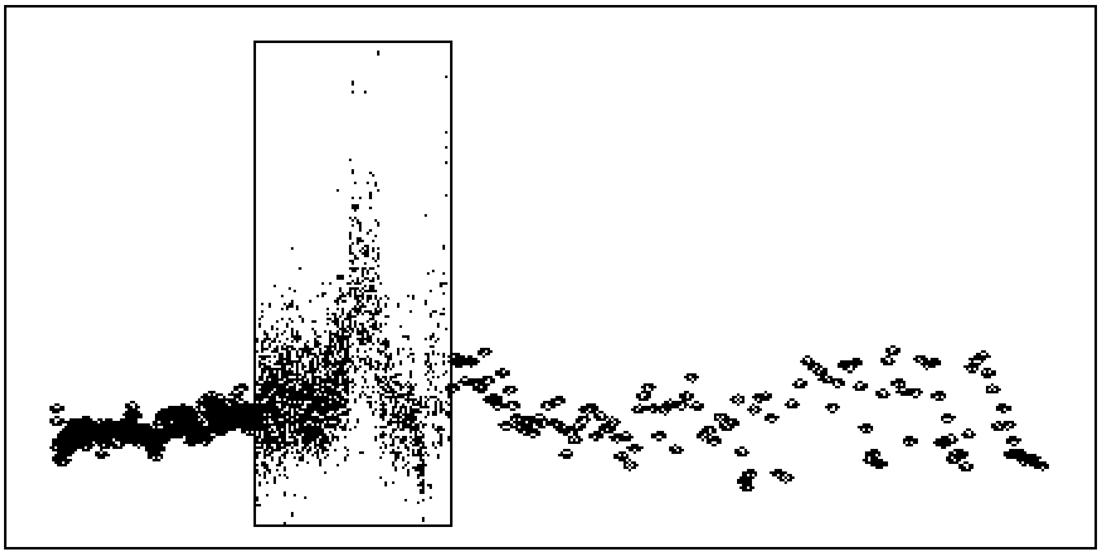

(b)

Figure 3.19: The wavelet brushing example [WB96]. (a) The brush data is defined. (b) Fine brushed data is painted over a coarse data background. This figure is used without the authors' permission.

3.3 Approximation Operators

Definition: *Approximation* in the context of the framework refers to a simplification in which complex relationships among objects are inexactly represented by simpler rela-

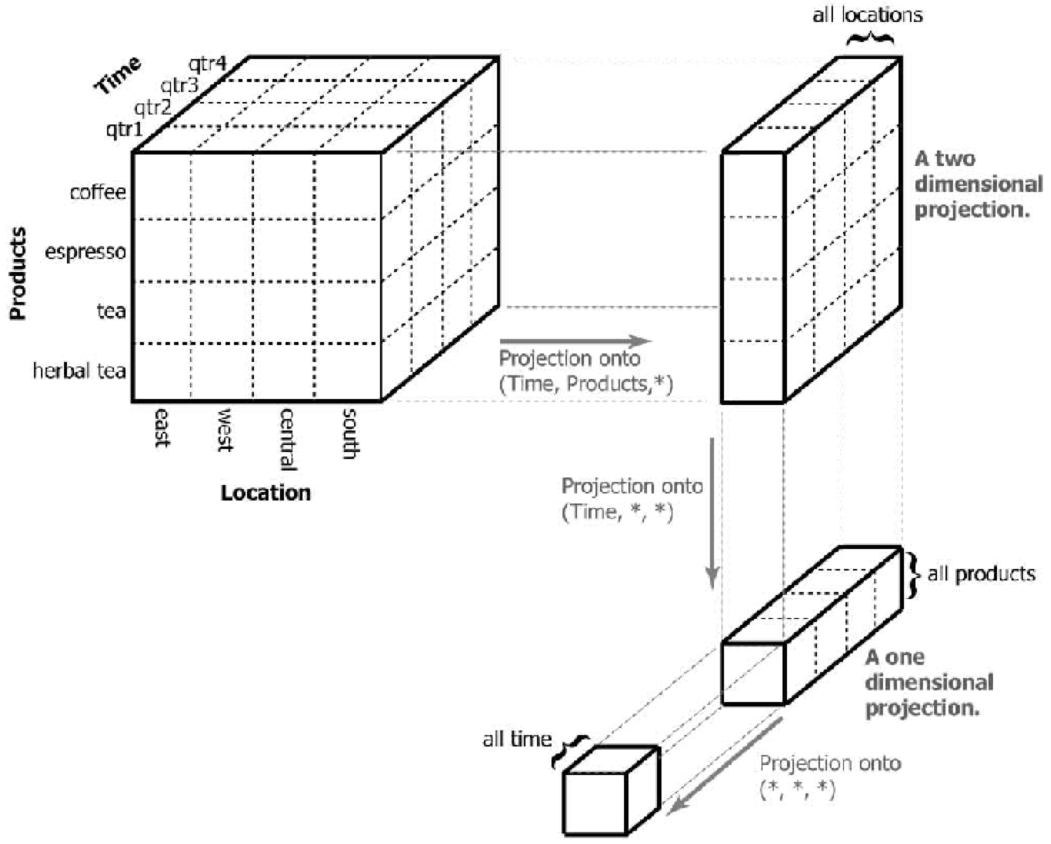
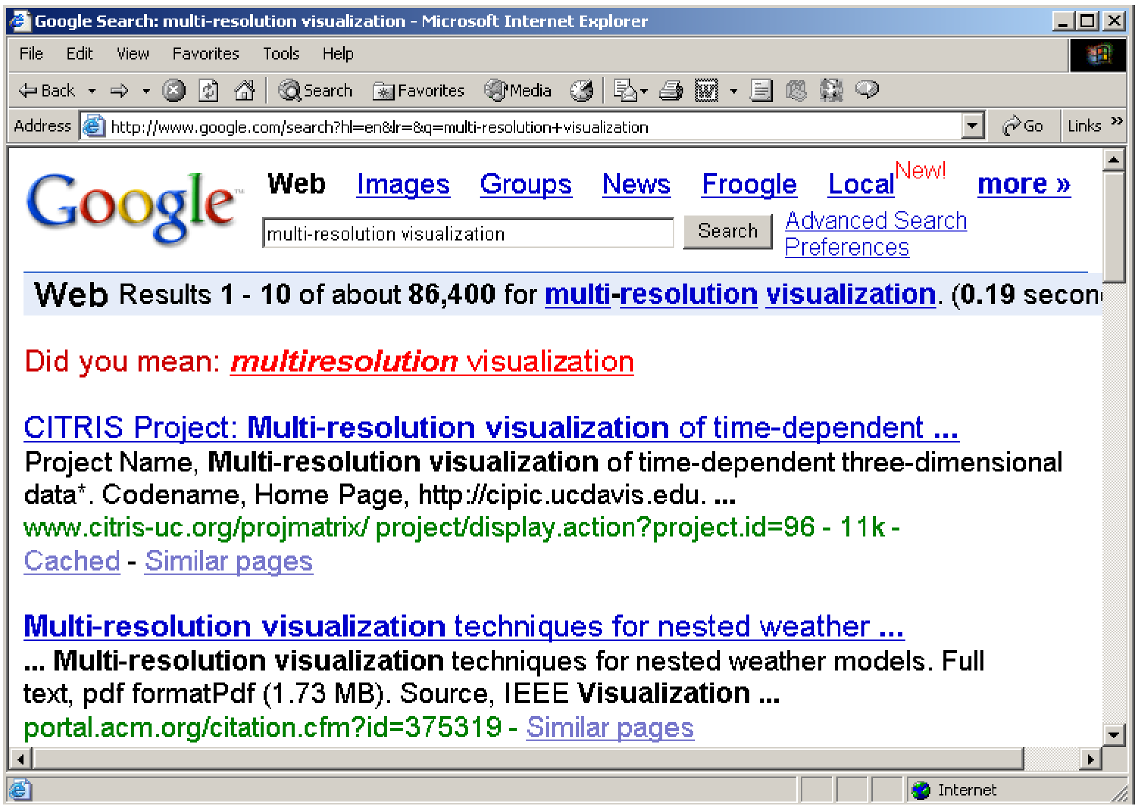

Time
qtr4
qtr3
qtr2
qtr1
coffee
espresso
tea
herbal tea
Products
east
west
central
south
Location
all locations
A two dimensional projection.
Projection onto (Time, Products,*)
Projection onto (Time, *, *)
all products
A one dimensional projection.
all time
Projection onto (*, *, *)

Figure 3.20: Projecting a three-dimensional data cube [SH00]. This figure is used without the authors' permission.

tionships. Approximation is an approach whose input is an object set $A = (a_1, a_2, ..., a_i)$ and whose output is another object set $B = (b_1, b_2, ..., b_i)$. For $1 <= j <= i$, b_j is an approximation of a_j. The relationship between each pair of objects in A can be estimated by measuring the relationship between their approximations in B. Objects in B have a different attribute set from objects in A.

Approximation operators include proximity positioning operators such as multi-dimension scaling and principal component analysis. There may exist other approximation operators besides proximity positioning operators.


Google Search: multi-resolution visualization - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://www.google.com/search?hl=en&lr=&q=multi-resolution+visualization
Go Links
Web Images Groups News Froogle Local New! more »
multi-resolution visualization
Search
Advanced Search
Preferences
Internet

Web Results 1 - 10 of about 86,400 for multi-resolution visualization. (0.19 secon

Did you mean: *multiresolution* visualization

CITRIS Project: Multi-resolution visualization of time-dependent ...
Project Name, **Multi-resolution visualization** of time-dependent three-dimensional data*. Codename, Home Page, http://cipic.ucdavis.edu. ...
www.citris-uc.org/projmatrix/ project/display.action?project.id=96 - 11k -
Cached - Similar pages

Multi-resolution visualization techniques for nested weather ...
... **Multi-resolution visualization** techniques for nested weather models. Full text, pdf formatPdf (1.73 MB). Source, IEEE **Visualization** ...
portal.acm.org/citation.cfm?id=375319 - Similar pages

Figure 3.21: In google, web pages in the result set are visually represented by sentences in the web-pages.

3.3.1 Proximity Positioning Operators

Definition: *proximity positioning* - if similarity between objects in an object collection can be measured, proximity positioning generates a topology preserving map of the collection that gives an overview of this proximity information. In the map, similar objects are positioned close to one another, and far from dissimilar ones [Bas01].

In the following sections, I introduce several popular proximity positioning approaches.

Multi-Dimensional Scaling

Multi-dimensional scaling (MDS) usually generates a two- or three-dimensional configuration of points - each representing a single object from a collection, with inter-point distances approximating the corresponding proximities [KW78].

A standard MDS analysis expresses the quality of this approximation as a loss function, and the optimal arrangement can be found by minimizing it numerically. It is known as least-squares metric MDS [Bas01].

Spring models [FR91] are another approach to MDS. They simulate physical forces to drive the layout process. In a spring model, each pair of objects is considered to have a spring, the ends of which are attached to the two points. The relaxed spring length is the ideal proximity of the two objects. Similar objects too far away are pulled together, and dissimilar objects too close together are pushed apart. The final layout produced by the system will reflect the spring system in equilibrium.

The computational cost of traditional MDS is very expensive and makes it unsuitable for large collections. Many MDS approaches have been proposed to scale to large collections. For example, incremental MDS [Bas99] considers only a carefully chosen subset of all pairwise proximities. Objects that make up cluster diameters at a certain level of the single link cluster hierarchy are identified, and are subject to traditional MDS, in order to establish the overall shape of the configuration. The remaining elements are positioned independently of one another with respect to this skeleton configuration. For very large collections, the skeleton configuration can itself be built up incrementally.

Other Proximity Positioning Approaches

Besides MDS, there are many other proximity positioning approaches. For example, Principal Component Analysis (PCA) [Jol86] attempts to project data down to a few dimensions that account for most variance within the data. Kohonen's Self Organizing Map (SOM) [Koh95, Fle99] is an unsupervised learning method to reduce multidimensional data to 2D feature maps [BCLC97]. Pathfinder network scaling [CNR96, MPM90] is a structural and procedural modeling technique which extracts underlying patterns in proximity data and represents them spatially in a class of networks called Pathfinder Networks.

3.4 Generalization Operators

Definition: *Generalization* is a simplification approach that sets one or more attributes of a group of objects to the same values. Its input is an object set $A = (a_1, a_2, ..., a_i)$. Objects in A have an attribute set $Attr = (var_1, var_2, ..., var_j)$. For $1 <= k <= j$, var_k is an attribute varies for different objects. Its output is the same object set with some variables in the attribute set becoming constants.

For example, all axes displayed in a parallel coordinates display can be generalized to have the same color. Users can change the color of all axes through a single color selection box.

Generalization is important in modern cartography. Cartographic generalization is a set of processes by which the content of a cartographic original is transformed into the content of a derived cartographic representation, so that they correspond perceptually with one another and with reality [BS02]. In other words, objects are reduced from individual to general and represented in a generalized form.

M. Eckert observed that the point of cartographic generalization was in selection and reducing to general [Eck21]. With large scale charts, the process of cartographic generalization first of all implies reduction of the content of cartographic originals. Changing to smaller scales, the reduction procedure moves to a selection procedure [BS02], so as to determine the kind of information to be presented on the chart, as it is not possible to present a large quantity of data on one sheet.

3.5 Discussion

In the above sections, a wide variety of simplification operators are discussed separately. Each large category of operators are defined as an approach whose input is a set of objects and whose output is another set of objects. In practice, these operators are often used in a mixed manner, i.e., the output set of a simplification approach can be used as the input set of another simplification approach. The second approach can be either the same kind of simplification as the first approach or a different kind of simplification. This process can be continued iteratively until the final output set is too small to be further simplified. For example, in the database management field, sampling and clustering are often used in a mixed manner.

Chapter 4

Simplification Operands and Spaces

A simplification operand is the section of space upon which a simplification operator is applied. A simplification operator can be applied to either structure S or language L in the KRA model. The former is called data space and the latter is called visualization space in the multi-resolution visualization framework proposed in this dissertation. In the following sections, I divide the data space and visualization space into smaller sub-spaces and discuss how simplification operators can be applied to these sub-spaces.

4.1 Data Space

Definition: *data* - digital representation of stimuli perceived from the world.

The data space corresponds to the second level of the KRA model, the *memorization of the perception* level, i.e., the structure S. The data structure used varies a lot in different visualization applications. I present a list of sub-spaces of the data space, i.e., data item space, dimension space, and topology space, and illustrate how simplification operators can be applied upon them using examples.

4.1.1 Data Item Space

Definition: *data items* - single pieces of data, such as individual rows in a data set stored in a table. In the data item space, data items are the objects in the input sets of the simplification operators.

Motivation

Simplification on the data item space has been widely studied. The reason is that data sets containing a large number of data items are very common nowadays and they clutter most traditional visualization techniques. For example, if every pixel on a 1024$*$1024 screen could present one data item, then the maximum number of data items we can visualize at the same time without overlap is 1,048,576. Unfortunately, large data sets nowadays easily exceed this size. Worse yet, most traditional visualization techniques have much lower screen usability than one pixel per data item. As a result, the clutter problem becomes a serious issue in the visualization of data sets with many data items. Figure 4.1 shows a data set containing 5 channels of remotely sensed data (SPOT, magnetics, and 3 radiometrics channels) visualized using the parallel coordinates visualization technique [ID90, Weg90]. Though this data set is only composed of 16,384 data items, the clutter problem is too serious to allow users to identify useful features of the data, such as clusters or anomalies.

Simplification Operators Applied upon Data Item Space

Simplification on the data item space can be found for almost all simplification operators. I will first give a detailed example of how to apply a hierarchical clustering operator upon the data items space to solve the clutter problem when visualizing data sets with a large number of data items and result in a the Interactive Hierarchical Display framework. Then I will give examples of how other operators are applied to the data item space in less

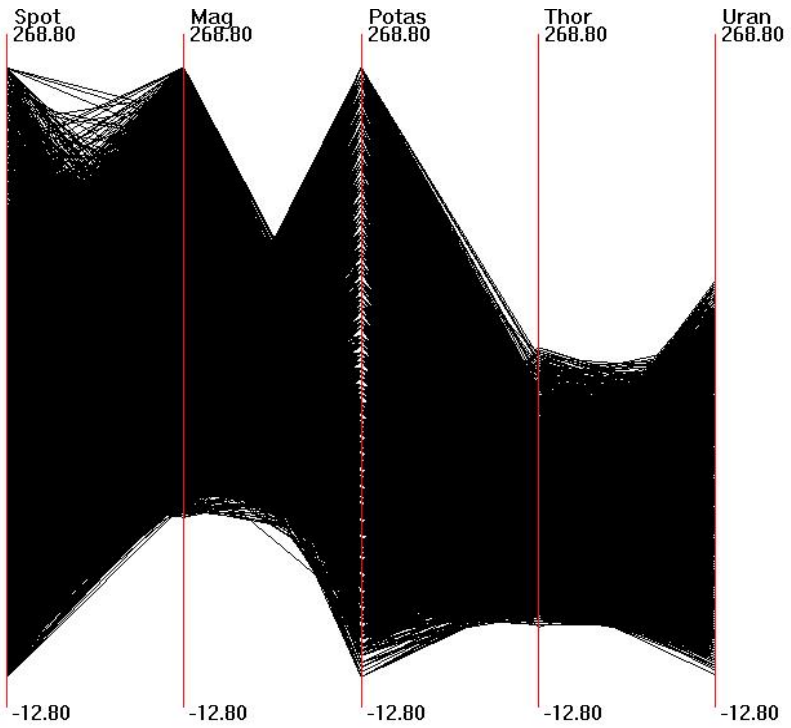


Figure 4.1: The clutter problem of the data item space [YWR03b]. A remotely sensed data set, obtained from Peter Ketelaar, CSIRO Division of Exploration and Mining (size: 16,384 data items, 5 dimensions) visualized with parallel coordinates.

detail.

The Interactive Hierarchical Display (IHD) framework was proposed as a general approach to interactively visualizing data sets with a large number of data items [YWR03b, YWR03a]. It was an extension of the thesis work by Ying-Huey Fua [Fua99]. IHD can be summarized as four steps:

- Step 1: Hierarchical Cluster Tree Generation

  First, a hierarchical cluster tree is constructed upon a data set according to similarity among the data items using a hierarchical clustering approach [GRS98, ZRL96, NH94]. Similar data items form a cluster, and similar clusters in turn compose higher-level clusters. A cluster is an abstraction of the data items composing it. It carries aggregate information of those data items, including:

– population: the number of data items contained in the cluster;

– range: the value range of the cluster in each dimension;

– mean: the numeric average value of the cluster in each dimension.

Each leaf node in the hierarchical cluster tree corresponds to an original data item and each non-leaf node corresponds to a data cluster. The collection of all leaf nodes presents exactly every data item of the data set, while the root is a cluster representing the whole data set. Users have the option of using the system-provided automatic clustering approach, using their customized clustering approaches to generate a hierarchical cluster tree, or specifying a hierarchical cluster tree manually. Figure 4.2 (a) illustrates how a two-dimensional data set containing ten data items are hierarchically clustered. Figure 4.2 (b) shows the resulted hierarchy. Figure 4.2 (c) shows how the IHD approach displays a hierarchy using a simple display where the shape of the tree is approximated.

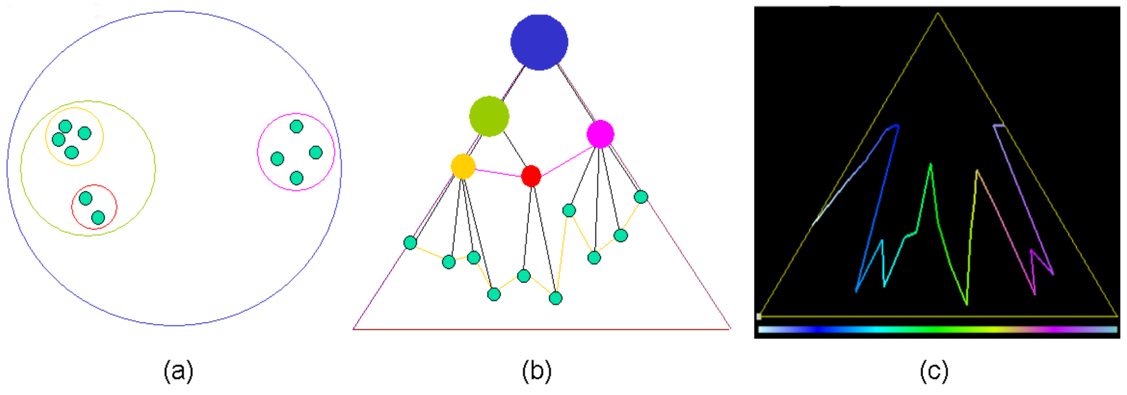


Figure 4.2: (a) Clustering. (b) Hierarchy generated. (c) Approximation display.

- Step 2: Hierarchical Cluster Tree Navigation and Brushing

  Next, the hierarchical cluster tree is visualized using a tool called a Structure-Based Brush (SBB) [FWR99a]. SBB is a brushing tool based on the approximate hierarchy display shown in Figure 4.2 (c). Through the SBB, users can interactively select region and levels of abstraction (LOAs) of interest from the tree. *Region* refers a set of leaf nodes in the tree. The region selected by the user is called the *brushed region*, while the regions outside the brushed region are called the *unbrushed regions*. *LOA* indicates the level of abstraction of a cluster or a set of clusters. They are expressed as levels, or sizes of the nodes in the tree. A region and an LOA work together to decide a set of nodes in the tree: the nodes of the indicated LOA containing all or any (depends on the user's selection) leaf nodes in the region as their direct or indirect children (please refer to [FWR99b, FWR99a] for more detail).

Using the SBB, users can define a brushed region, and assign an LOA for it and another LOA for the unbrushed region. The LOA for the unbrushed region is usually higher than the LOA for the brushed region, so that users can examine the brushed region in detail with the unbrushed region in a higher level of abstraction as context information.

- Step 3: Cluster Fetching

  In this step, nodes in the tree selected by the brushed/unbrushed regions and their LOAs are accessed and the aggregate information of the data clusters carried by them are fetched. When the tree is large and stored in a database, queries have to be used in order to fetch the data. Some advanced techniques, such as caching and prefetching, can be used to improve the speed of this process [Str00, DRW03, SRW00, DRRW03].

- Step 4: Cluster Visualization Using the Mean-Band Method

  Finally, the aggregate information of the selected data clusters is visualized using extensions of existing multi-dimensional visualization techniques. A technique named the *mean-band method* is applied to traditional multi-dimensional visualization techniques in order to convey the aggregate information of data clusters. Figure

> 3.9 gives an example of the mean-band method. Using the mean-band method, the mean of a data cluster is depicted as if it is a data item in the data space and visualized using traditional multi-dimensional visualization techniques. The extents of the data cluster are depicted as a hyper-box surrounding the mean and named a band, which has a varying opacity that decreases linearly from the cluster center to the cluster edges as a rough approximation of the density of the data cluster. The opacity is set to a small value at the edges, and set proportional to the cluster population at the center so that large clusters are more readily discerned than small ones.
>
> The mean-band method uses *proximity-based coloring* [FWR99b, FWR99a] to convey the structural relationships among the data clusters in the SBB and the data cluster displays and link the hierarchical cluster tree with the data cluster displays. It assigns colors to clusters based on the structure of the tree and uses the same color to depict the node in the SBB and the data cluster displays,
>
> The mean-band method is independent from individual multi-dimensional visualization techniques. It is a general method that can be applied to most existing multi-dimensional visualization techniques so that they can be combined into the IHD framework. Till now it has been applied to parallel coordinates, star glyphs, scatterplot matrices, and dimensional stacking in XmdvTool. Figures 4.3 - 4.5 show the extensions of parallel coordinates, scatterplot matrices, and star glyphs in the IHD framework.

[CK03] presents an approach to visualize results of microarray experiments that record the expression of thousands of genes using a hierarchical clustering operator. The first stage of clustering is the creation of a similarity matrix. In the similarity matrix both rows and columns correspond to the full list of genes - cells reflect inter-time-series similarities

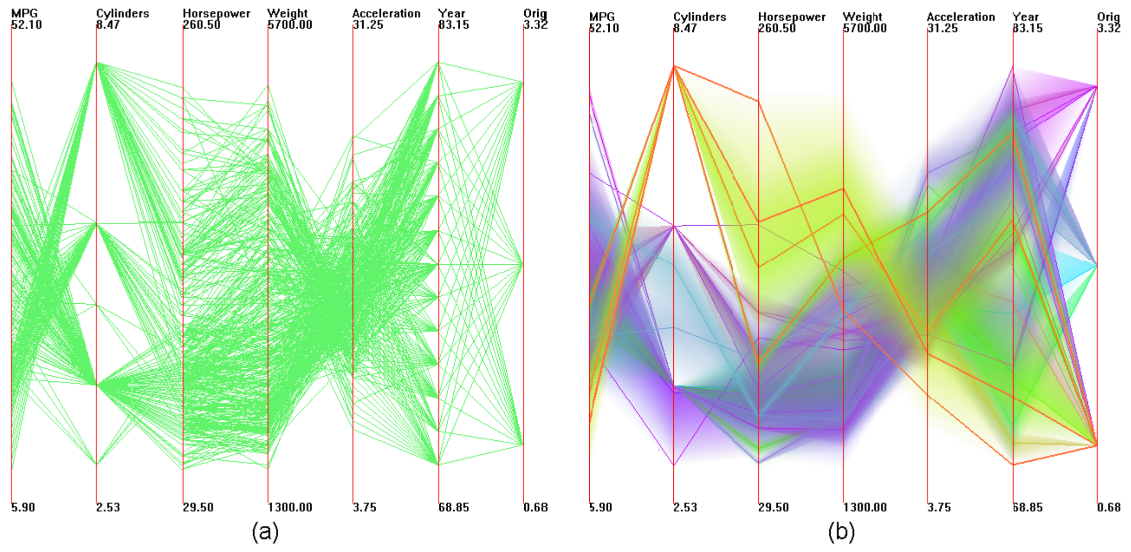


Figure 4.3: The Cars data set. (a) Traditional parallel coordinates. (b) Parallel Coordinates in the IHD framework.

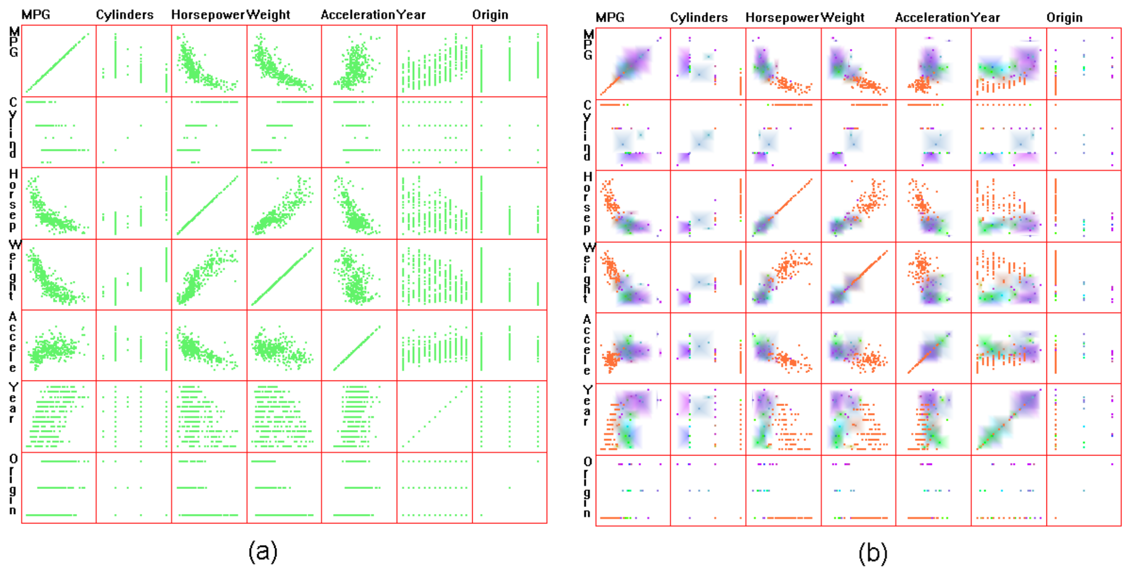


Figure 4.4: The Cars data set. (a) A traditional Scatterplot Matrix (b) A Scatterplot Matrix in the IHD framework.

calculated according to some predefined similarity measures [Qua01]. Then a hierarchical agglomerative clustering [ESBB98, SS02] is used upon the similarity matrix to produce a binary tree known as a dendrogram. In Figure 4.6 [CK03], an example dendrogram is visualized. The end points of outer branches of the dendrogram represent genes with the

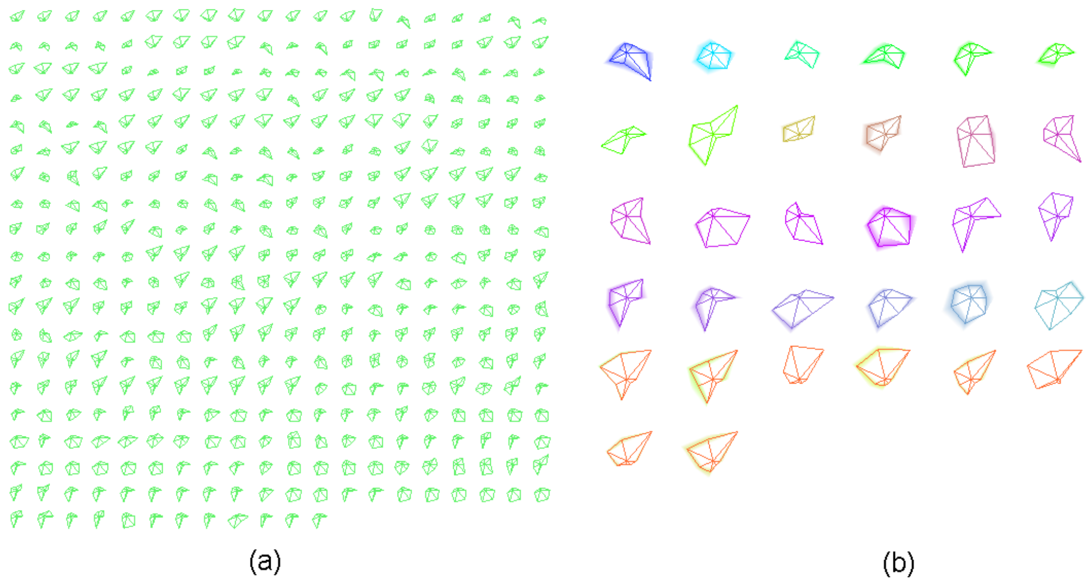


Figure 4.5: The Cars data set. (a) Traditional star glyphs (b) Star glyphs in the IHD framework.

similarity groupings represented by common branches in the tree structure [CK03]. In this display gene expression patterns are color-coded and stacked vertically beside their respective gene nodes.

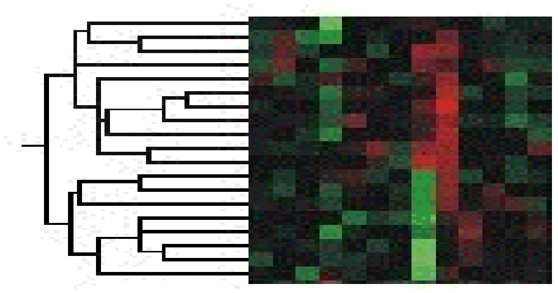
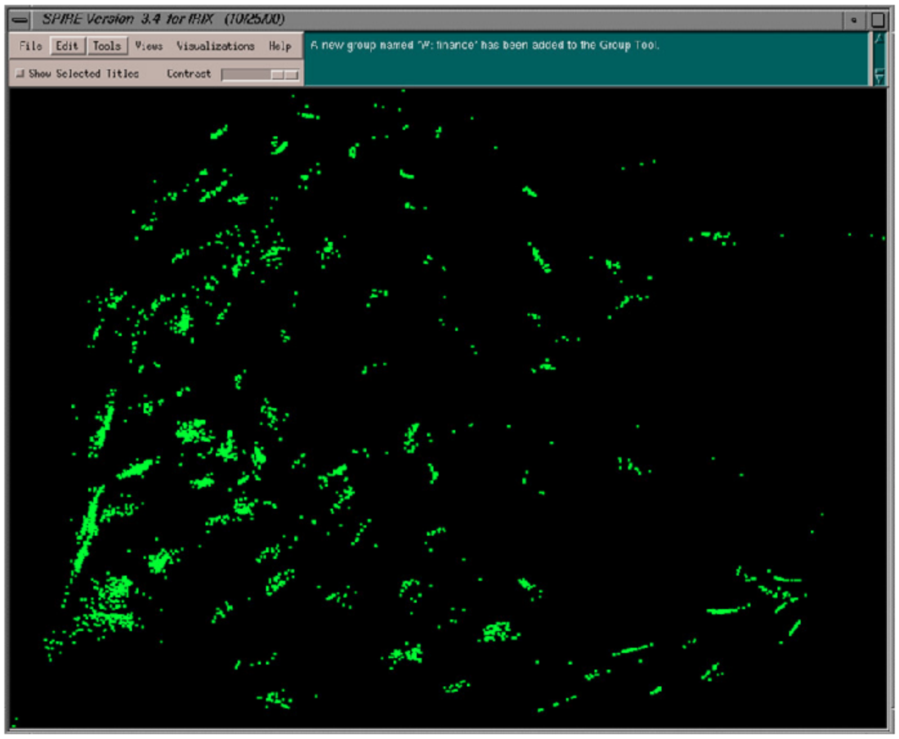


Figure 4.6: A microarray data set. A clustering operator is applied upon the data item space to generate the dendrogram. The gene expression patterns are color-coded [CK03]. This figure is used without the authors' permission.

The SPIRE Galaxies visualization [WTP$^+$95], which uses a proximity positioning operator to visualize large collections of documents, is reminiscent of the night sky. Figure 4.7 [NHT01] shows a document collection visualized in a SPIRE Galaxies display. This visualization represents each document as a small green dot; the collection as a whole looks like a universe of “docustars”. Closely related documents cluster together in a tight group while unrelated documents are separated by large space. The distance between points indicates their thematic similarity. Thus, if points in the visualization are close together, then it is likely that the corresponding documents contain thematically similar information. If they are far apart, the documents probably are thematically diverse. For the users, it seems that related documents are automatically sorted into the correct piles.



A new group named "W: finance" has been added to the Group Tool.

Figure 4.7: A document collection visualized in the SPIRE Galaxies visualization [NHT01]. This figure is used without the authors’ permission.

The Beneath Highfields visualization [KLS00] applies a self-organizing map (SOM) operator upon the data item space. In Figure 4.8, the Cars data set is visualized in a Beneath Highfields display. It shows a self-organizing map created upon the data items in the data set. Each peak in the map displays a cluster of similar objects. The number of objects within a single cluster is mapped onto the height of the peak. Color is used for conveying similarities between adjacent clusters where bright intensities denote a higher degree of dissimilarity.

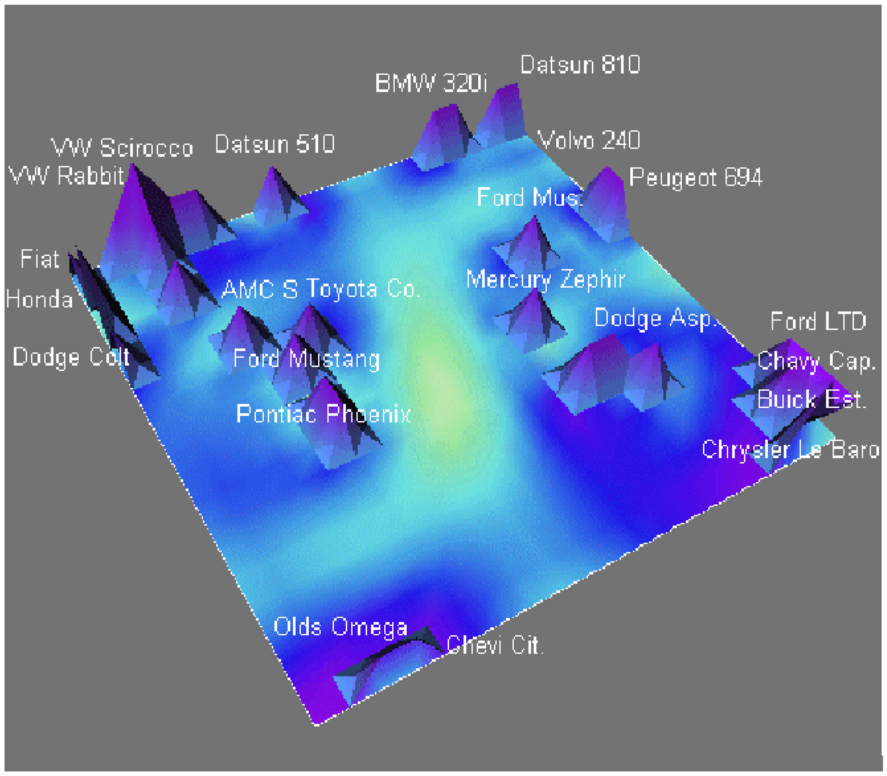


Figure 4.8: The Cars data set is visualized using the Beneath Highfields visualization [KLS00].

In addition, the Astral Telescope Visualiser [DE02] discussed in Chapter 3 is an example of applying the sampling operator upon the data item space.

Special Cases

In the above examples, the simplification operators are applied to the data item space considering all attributes of the data items. However, there are often special cases where only one or a few special attributes of the data items are considered in simplification. These special attributes are often the time dimension for a time-series data set, the geometry dimensions for a geometry data set, or independent variables when there are both independent and dependant variables in the data set. Simplification in these special cases is often simpler than the general cases, since fewer attributes need to be considered in the special cases.

For example, SolarPlot [Chu98] simplifies a time-series data set using an Equi-width one-dimensional histogram operator. The time dimension is visualized using a circumference and a variable depending on time is mapped to the lengths of spokes starting from pixels in the circumference corresponding to different time values. When there are fewer pixels in a circumference than the discrete time values, each pixel corresponds to a time period, and the length of the spoke starting from it is the sum of the values of the dependant variable at all discrete time values within that time period. Thus an aggregation operator is used in this approach. Figure 4.9 shows ticket sales data for a hypothetical movie theatre over a period of 30 years in SolarPlot [Chu98]. The circumference encodes the time in which a particular ticket was sold (time starts from 0 degree, which is at the mid-bottom point of the circle, to 360 degrees anti-clockwise). The lengths of the strokes encode the number of tickets sold in corresponding time periods. In Figure 4.9 (a), a small circle is used and the ticket sales data is highly aggregated. It can be seen that more tickets are sold in the later dates than in the earlier dates. In Figure 4.9 (b), a larger circle is used and the data is less highly aggregated. More detail of the ticket sales data is revealed.

4.1.2 Dimension Space

Definition: *dimensions* - individual measures of data items, such as individual columns, or variables, of a data set stored in a table. In the dimension space, dimensions are the objects in the input sets of the simplification operators.

The dimension space is simple for a data set that contains only a few dimensions. However, high dimensional data sets are becoming commonplace in applications such as digital libraries, bioinformatics, simulations, process monitoring, and surveys. It is no

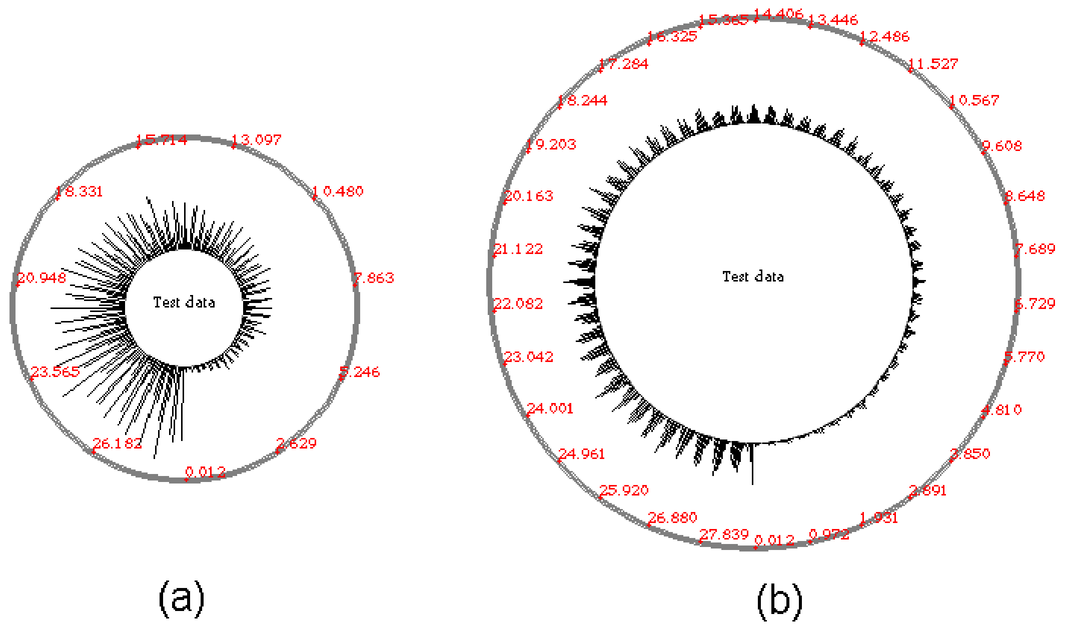


Figure 4.9: Ticket sales data in SolarPlot [Chu98]. (a) Highly aggregated data (b) Less highly aggregated data. This figure is used without the authors' permission.

longer unusual to have data sets with hundreds or even thousands of dimensions. Traditional visualization techniques for multidimensional data sets, such as glyph techniques [And72, SFGF72, Che73, RAEM94], parallel coordinates [ID90, Weg90], scatterplot matrices [CM88], and pixel-level visualization [KKA95], do not scale well to high dimensional data sets. For example, Figure 4.10 shows a subset of the Census Income data set [HB99], which has 42 dimensions and 200 data items. While the number of data items in this display is not large, individual data items cannot be seen clearly from this display since the number of dimensions is large. A large number of axes crowd the figure, preventing users from detecting any patterns or details. Even with low numbers of data items, high dimensionality presents a serious challenge for current display techniques.

To overcome the clutter problem, a simplification approach frequently described in the literature is to automatically approximate the large number of dimensions using a few dimensions (usually 2 or 3). This approach is called *dimensionality reduction*. The idea

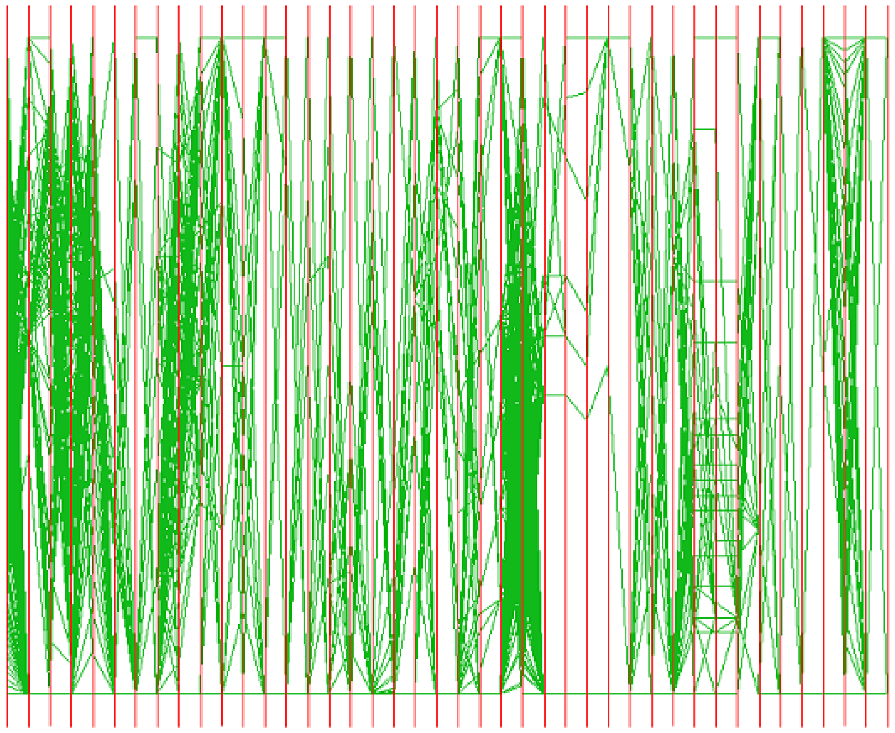


Figure 4.10: A subset of the Census Income data set (42 dimensions, 200 data items) in Parallel Coordinates [YWRH03]. Individual data items cannot be seen clearly.

is to first reduce the dimensionality of the data while preserving the major information it carries, and then visualize the data set in the reduced dimensional space. There are several popular dimensionality reduction techniques used in data visualization, including Principal Component Analysis (PCA) [Jol86], Multidimensional Scaling (MDS) [Mea92], and Kohonen's Self Organizing Maps (SOM) [Koh95, Fle99]. These automatic approaches usually take a high dimensional data set as input and produce the projections of the data set in a two-dimensional or 3-dimensional space as output. These approaches can be categorized into applications where the proximity positioning operators are applied upon the data item space. However, from the perspective of the dimension space, they can also be viewed as applications where approximation operators are applied upon the dimension space. Only two levels of detail are generated in the dimension space: the original high dimensional space is the finer view, and the projected two-dimensional or 3-dimensional space is the coarser view. Little user interaction is allowed in these approaches.

Other simplification operators can be applied upon the dimension space. For example, Random Mapping [Kas98] projects the high dimensional data to a lower dimensional space using a random transformation matrix for clustering. Using this approach, multiple lower dimensional spaces with the same number of dimensionality and lower dimensional spaces with different dimensions can be generated.

I have conducted an extended study on simplification in the dimension space, and proposed several multi-resolution visualization approaches for visualizing high dimensional data sets. In the following sections, I present these approaches in detail.

Visual Hierarchical Dimension Reduction Framework

The Visual Hierarchical Dimension Reduction (VHDR) framework [YWRH03] applies a hierarchical clustering operator upon the dimension space. It was proposed as a general approach towards interactively visualizing high dimensional data sets. Figure 4.11 shows the system structure of the VHDR framework [YWRH03]. The methodology can be divided into five steps:

- Step 1: Dimension Hierarchy Generation

  First, all original dimensions of a multidimensional data set are organized into a dimension hierarchy according to similarities among the dimensions. Dimensions are said to be similar if the majority of data items have similar values on them. Each original dimension is mapped to a leaf node in the hierarchy. Similar dimensions are placed together to form a cluster, and similar clusters in turn compose higher-level clusters. Users have the option of using the system-provided automatic clustering approach, using their customized clustering approaches, or specifying a dimension hierarchy manually.

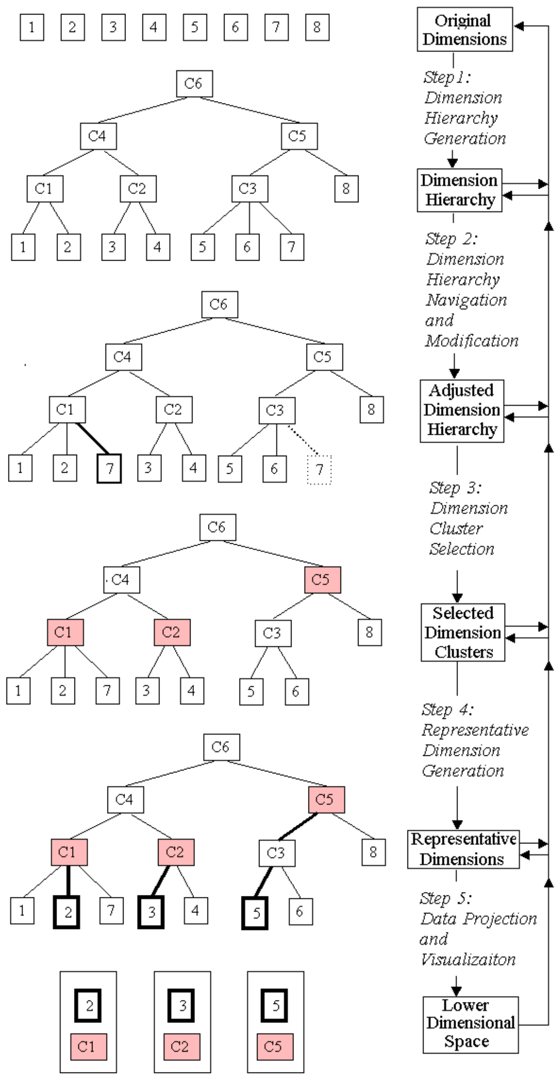
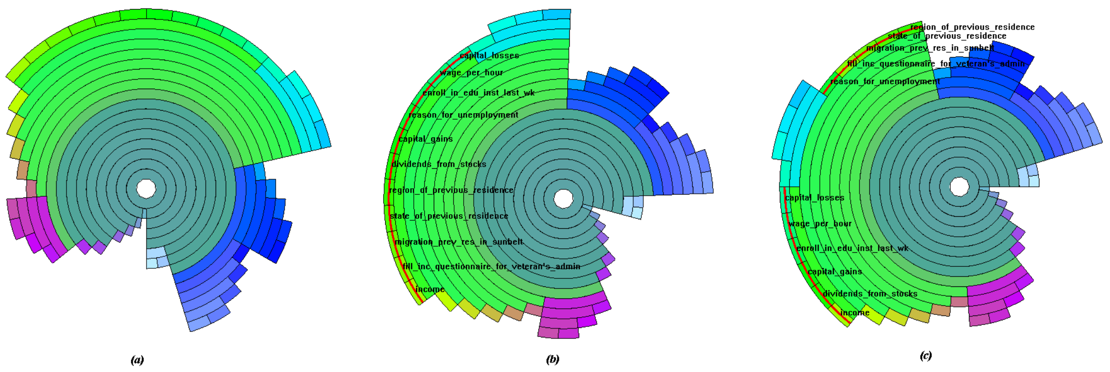

Original Dimensions
Step1: Dimension Hierarchy Generation
Dimension Hierarchy
Step 2: Dimension Hierarchy Navigation and Modification
Adjusted Dimension Hierarchy
Step 3: Dimension Cluster Selection
Selected Dimension Clusters
Step 4: Representative Dimension Generation
Representative Dimensions
Step 5: Data Projection and Visualizaiton
Lower Dimensional Space
C1
C2
C3
C4
C5
C6

Figure 4.11: System Structure of VHDR [YWRH03].

- Step 2: Dimension Hierarchy Navigation and Modification

  Next, users can navigate through the dimension hierarchy in order to gain a better understanding of it. Users can also interactively modify the hierarchy structure and supply meaningful names to the clusters. The dimension hierarchy is visualized in a radial space-filling display named InterRing, which contains a suite of navigation and modification tools. Figure 4.12 shows the dimension hierarchy of the Census data set in InterRing. More detail of dimension hierarchy navigation and modification can be found in Chapter 5.



capital_losses
wage_per_hour
enroll_in_edu_inst_last_wk
reason_for_unemployment
capital_gains
dividends_from_stocks
region_of_previous_residence
state_of_previous_residence
migration_prev_res_in_sunbelt
fill_inc_questionnaire_for_veteran's_admin
income
region_of_previous_residence
state_of_previous_residence
migration_prev_res_in_sunbelt
fill_inc_questionnaire_for_veteran's_admin
reason_for_unemployment
capital_losses
wage_per_hour
enroll_in_edu_inst_last_wk
capital_gains
dividends_from_stocks
income
(a)
(b)
(c)

Figure 4.12: Dimension hierarchy of the Census data set in InterRing. (a) The automatically generated hierarchy (b) The detail of a cluster is revealed after brushing and rotation. (c) The hierarchy is modified by moving some dimensions from the cluster examined in detail in (b) to elsewhere [YWRH03].

- Step 3: Dimension Cluster Selection

  Next, users interactively select interesting dimension clusters from the hierarchy in order to construct a lower dimensional subspace. Several selection mechanisms are provided in InterRing to facilitate dimension cluster selection. Selected clusters are highlighted. More detail on dimension hierarchy selection can be found in Chapter 5.

- Step 4: Representative Dimension Generation

  In this step, a representative dimension (RD) is assigned or created for each selected dimension cluster. The selected dimension clusters construct the lower dimensional space through these RDs. RDs are selected to best reflect the aggregate characteristics of their associated clusters. For example, an RD can be the average of all the original dimensions in the cluster, or can be an original dimension located in the center of the cluster. Users have the option of selecting one of the system-provided RD generation methods or using a customized one.

- Step 5: Data Projection and Visualization

  Finally, the data set is projected from the original high dimensional space to a lower dimensional space (LD space) composed of the RDs of the selected clusters. We call its projection in the LD space the *mapped data set*. The mapped data set can be viewed as an ordinary data set in the LD space and can be readily visualized using existing multidimensional visualization techniques. This is an advantage of VHDR; it is so flexible that it can be applied to any existing multidimensional data visualization technique. In order to provide further dimension cluster characteristics in the LD space, such as the dissimilarity information between dimensions within a cluster, we attach the dimension cluster characteristics information to the mapped data set and provide the option to display it using extensions to the data visualization techniques.

Users can return to any of the previous steps at any time to refine the process iteratively. For example, after the LD space has been generated, users can still modify the dimension hierarchy, redo the selection, and generate a different LD space.

Figure 4.13 gives examples of the VHDR approach. Figure 4.13 (a) shows a 42 dimensional data set in parallel coordinates (the left view) and its dimension hierarchy (the right view). Figure 4.13 (b) shows that three dimensions in the same cluster are selected from the dimension hierarchy and the corresponding lower dimensional space is displayed in parallel coordinates. This lower dimensional space is composed of closely related dimensions. In Figures 4.13 (c), dimensions and dimension clusters covering all original dimensions in the hierarchy are selected to form a lower dimensional space that reveals more variance of the data set.

One significant advantage of the VHDR approach is that the generated lower dimensional space is meaningful to users, in that:

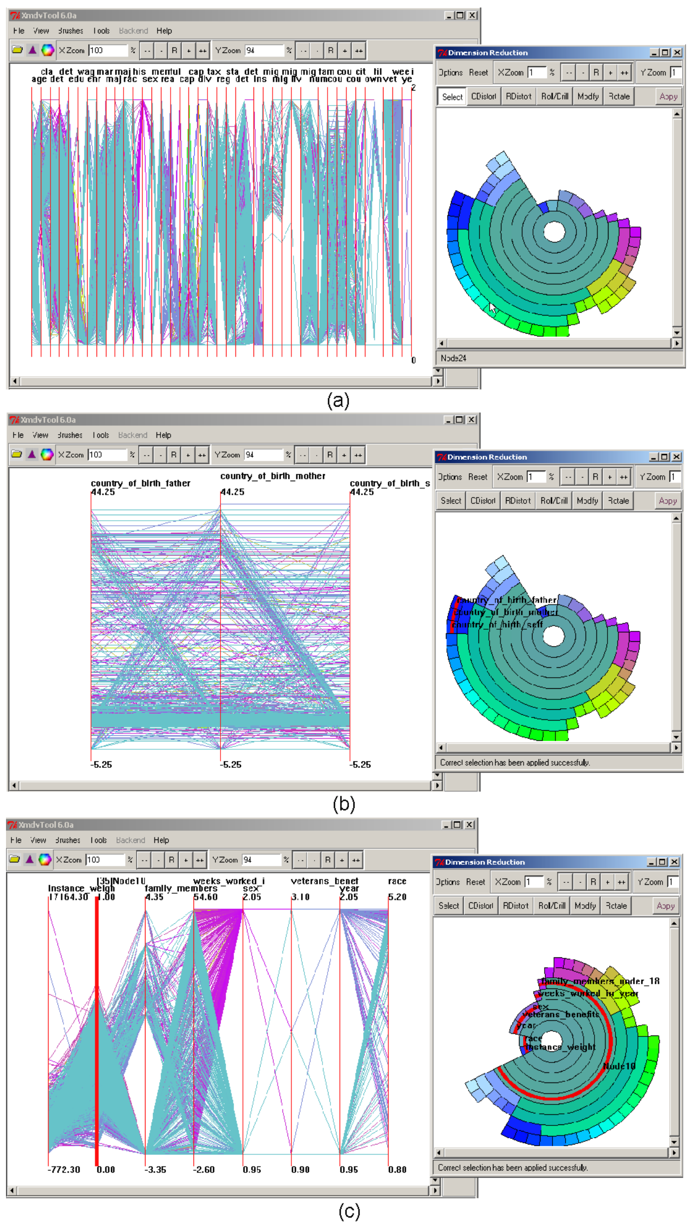


Figure 4.13: VHDR examples. (a) The original data set and its dimension hierarchy (b) A lower dimensional space composed of closely related dimensions is constructed. (c) A lower dimensional space covering major variance of the data set is constructed.

- As long as users understand the similarity measure used in dimension clustering, the meaning of the dimension hierarchy is straightforward.

- Through dimension hierarchy visualization and navigation, if users find that a cluster contains some dimensions that have no semantic relationship with other dimensions in the cluster according to their knowledge of the data domain, they can manually remove them from this cluster. Similarly, they can merge two clusters that are semantically related. Thus each cluster can have a clear domain-specific meaning guided by a domain expert.

- Users can label the dimension clusters with “meaningful” names, thus helping the interpretation of the dimension clusters during later data exploration.
- Users interactively select the dimension clusters to be visualized in the LD space according to their knowledge of the data domain. As a result, the structure of the LD space is meaningful to them.
- Users can select the RD generation approach or even explicitly select the RDs, hence they know how the RDs are generated and how to interpret them.

Visual Hierarchical Dimension Filtering

The Visual Hierarchical Dimension Filtering (VHDF) [YPWR03] approach applies a sampling operator on the dimension space. The inspiration of VHDF is that a high dimensional data set often contains many similar or even nearly identical dimensions. Removing redundant dimensions from the display reduces the cluster problem while at the same time retaining major variations of the data set. In addition, dimensions that are not important for the users can also be removed without affecting the users performance of their exploratory tasks. The importance of each dimension is decided by a user’s particular visualization task. For example, if the user is looking for variance of a data set, a dimension that contributes more to the variance of the data set is more important than a dimension that contributes less to the variance. To this end, the importance value of a dimension is decided by the contribution of it to the variance. The importance value of a dimension cluster is an aggregation of its children. Users can define their own importance measures.

Similar to the VHDR approach, VHDF is based on a dimension hierarchy generated using a hierarchical dimension clustering approach. But each node is assigned an importance value in the VHDF approach. In addition, only leaf nodes in the hierarchy can be sampled. Dimensions to be displayed are sampled according to a filtering criterion that is a combination of the dimension similarity and importance. It is assumed that if some dimensions are very similar to each other, then only one of them should be left in the display. It is also assumed that if some dimensions are unimportant for a user's visualization task, they should not be displayed. Thus a similarity threshold and an importance threshold are used in the filtering. The algorithm to select dimensions for display is a recursive approach starting from the root of the dimension hierarchy.

For each node, its importance value is examined. If it is smaller than the importance threshold, ignore the node and return. That is, the dimensions contained in unimportant nodes are ignored. If it is larger than the importance threshold and it is a leaf node, select it and return. Otherwise, check the similarity among the dimensions in the node. If it is larger than the similarity threshold, apply the approach on its immediate descendants. Otherwise, only apply the approach on its most important immediate descendant.

Figures 4.14 and 4.15 give examples of dimension filtering. Figure 4.14 shows four data items in the OHSUMED data set before and after dimension filtering in Star Glyphs (all dimensions are assigned the same importance value). Comparing Figures 4.14 (a) with 4.14 (b) (dimensions in Figure 4.14 (b) have the same order as in Figure 4.14 (a)), the shapes of the corresponding glyphs seem to be significantly related. In the filtered display, the number of dimensions is much more manageable compared to the unfiltered ones. Figure 4.15 shows the Scatterplot Matrix display of the OHSUMED data set before and after dimension filtering. In Figure 4.15 (a), individual plots cannot be discerned without significant zooming. In Figure 4.15 (b), the number of plots has been greatly reduced.

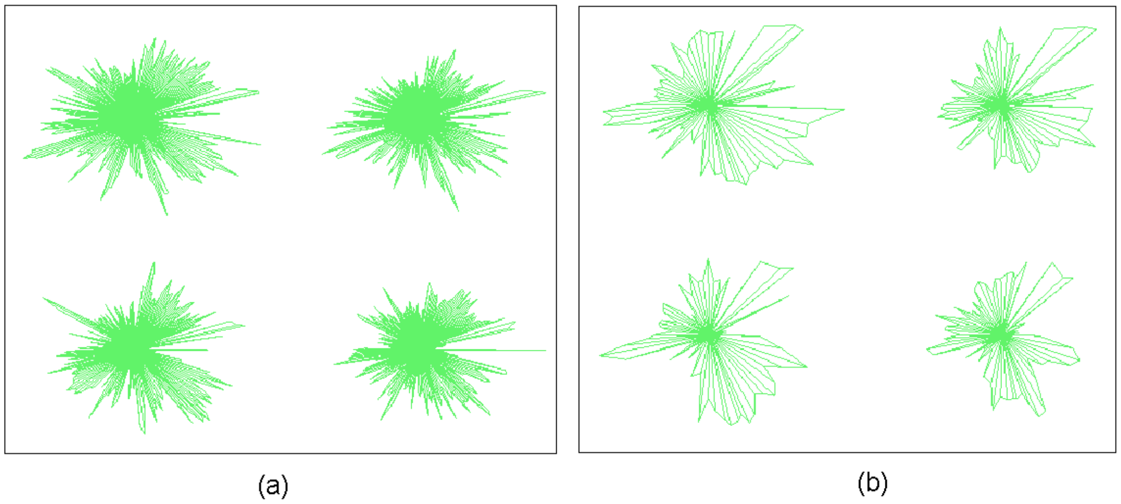


Figure 4.14: Dimension filtering in star glyphs [YPWR03]. Four data items in the OHSUMED data set are shown. (a) Before filtering (b) After filtering.

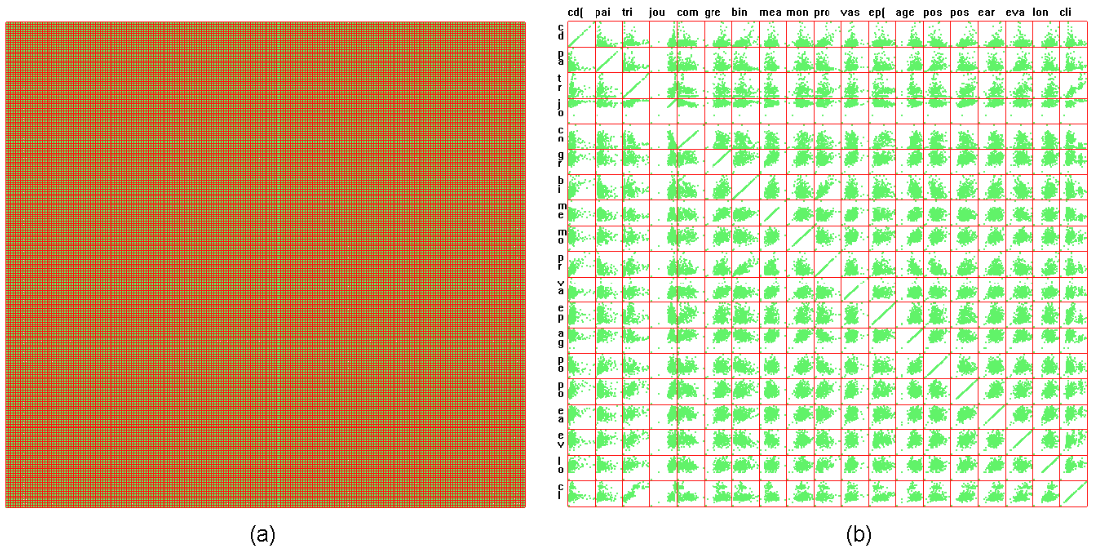


Figure 4.15: Dimension filtering in scattterplot matrices [YPWR03]. The OHSUMED data set is shown. (a) Before filtering (b) After filtering.

Value and Relation Displays

Value and Relation (VaR) displays [YPH+04] apply a Multi-Dimensional Scaling (MDS) operator on the dimensions (rather than on the data items, as in other dimension reduction approaches) to explicitly convey the relationships among the dimensions. VaR displays aim to interactively visualize data sets containing up to hundreds or thousands of dimensions without dimension reduction.

By graphically presenting each dimension of a high dimensional data set as a glyph in a 2D space, the VaR display conveys relationships such as correlation among the dimensions through the positions of the glyphs. The positions of the glyphs are generated using Multi-dimensional Scaling (MDS) [KW78] according to the pair-wise relationships among the dimensions. In Figure 4.16 (a), each dimension of the SkyServer data set (361 dimensions, 50,000 data items) is mapped to a dot and positioned in the 2D space using MDS. Such a display is called a *star field* display. In this example, the correlation among the dimensions is used to generate the positions through MDS. Thus closely related dimensions have positions adjacent to each other in this display. It reveals the correlation among the dimensions intuitively.

Besides the relationships among the dimensions, the VaR display conveys values of the data items using pixel-oriented techniques [KKA95]. Pixel-oriented techniques are visualization methods that map values of data items to the color of pixels and arrange pixels to convey relationships. We use pixel-oriented techniques to map values of the data items within a single dimension to pixels and arrange them into a *glyph* ("subwindow", as termed in other papers on pixel-oriented techniques [Kei00]). For each dimension, a glyph can be generated. We replace the dots in the star field display by their respective glyphs to produce the VaR display. Figure 4.16 (b) shows the VaR display of the SkyServer data set generated by replacing the dots in Figure 4.16 (a) by glyphs. Figure 4.16 (c) shows an enlarged glyph. The textures of the glyphs reveal data patterns in the dimensions.

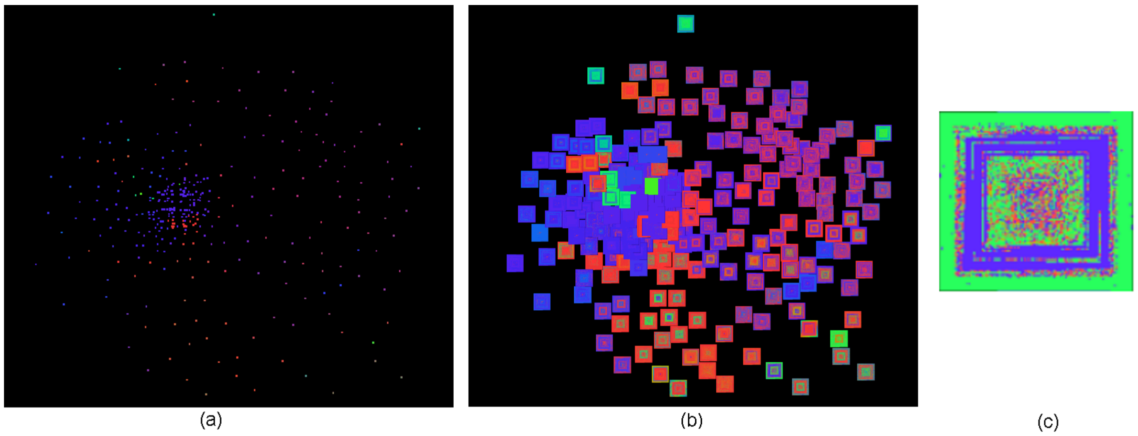


Figure 4.16: The VaR Display [YPH+04]. (a) A star field display where each dimension is mapped to a dot and positioned using MDS according to the correlation among the dimensions. (b) The dots in (a) are replaced by glyphs that present values of the data items to form a VaR display. (c) An enlarged Glyph. The data set is the SkyServer data set (361 dimensions, 50,000 data items), which was extracted from the Sloan Digital Sky Server (SDSS) data [GST+01].

A rich set of navigation and selection tools has been provided for the VaR displays. They will be discussed in Chapter 5.

4.1.3 Topology Space

Definition: *topology* - geometric, spatial, temporal, or logical relationships among data items or dimensions of a data set. For example, the links among the nodes in a graph form the topology space of the graph data set. In the topology space, the relationships are the objects in the input sets of the simplification operators.

Simplification in the topology space is also raised by the clutter problem of visualizing large data sets containing complex topologies. For example, Figure 4.17 (a) shows a graph that contains 87,931 vertices and 87,930 edges. The graph is so cluttered than most vertices and edges cannot be seen.

Various simplification operators have been applied upon the topology space to generate multi-resolution visualization. For example, [GKN04] uses a hierarchical clustering

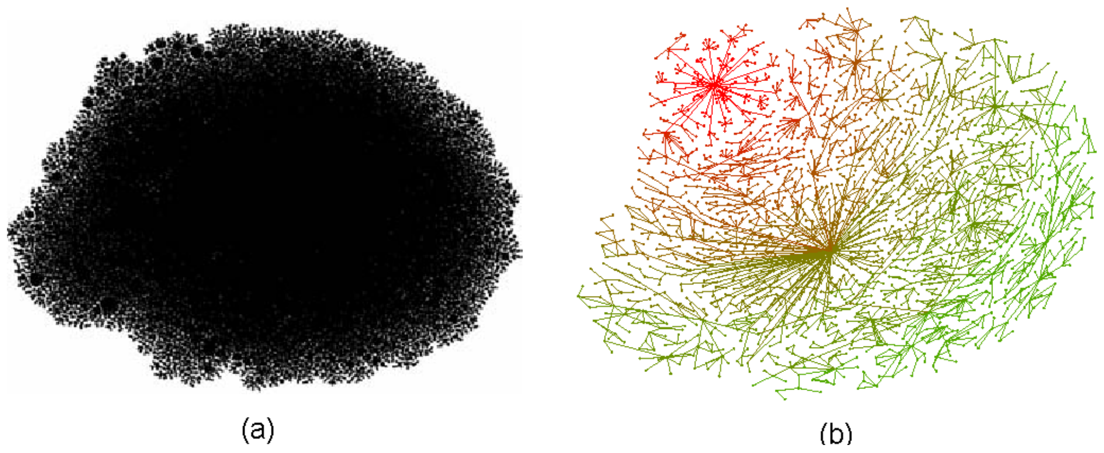


Figure 4.17: A graph that contains 87,931 vertices and 87,930 edges [GKN04]. (a) The display is seriously cluttered. (b) The same graph displayed in lower levels of detail. The major structure of the graph is revealed in the green part while the red part shows detail of a part of the graph. This figure is used without the authors' permission.

approach to create a multi-resolution graph visualization. Figure 4.18 illustrates how a graph is hierarchically clustered to form a hierarchy. Figure 4.19 shows how a coarse graph is created by selecting nodes from the hierarchy. Figure 4.17 (b) shows the same data set which is shown in Figure 4.17 (a) in lower levels of detail. Within the display, the red part of the graph has a higher level of detail than the green part. The major structure of the graph is revealed in the green part while the red part shows detail of a part of the graph.

The narrative flow of a document is the topology space of the document. Without simplification, a user needs to read a document in order to learn the narrative flow of it. The TOPIC ISLANDS visualization system [MWBF98] applies a wavelet transform operator upon the topology space of a document to abstract the narrative flow so that it can be visualized in a succinct way. Its underlying technique, called TOPIC-O-GRAPHY, applies wavelet transforms to a custom digital signal constructed from words within a document. The resultant multi-resolution wavelet energy is used to analyze the characteristics of the narrative flow in the frequency domain. Based on the energy changes within the narrative

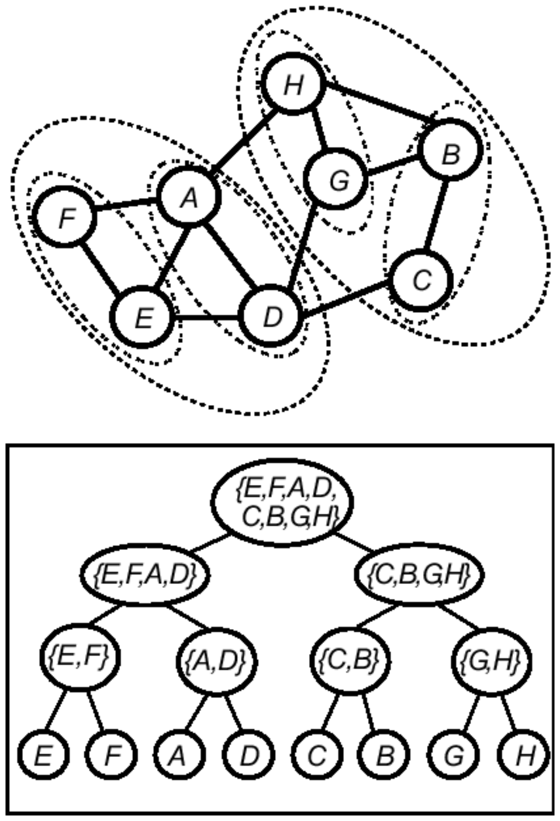


Figure 4.18: The illustration of how a graph is hierarchically clustered to form a hierarchy [GKN04]. This figure is used without the authors' permission.

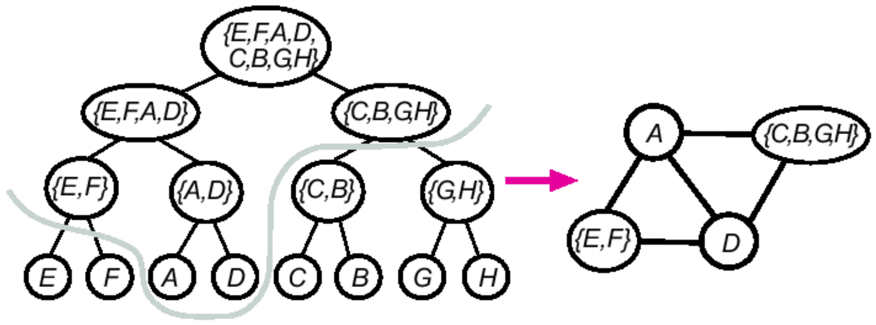


Figure 4.19: The illustration of how a coarse graph can be obtained from a hierarchy [GKN04]. This figure is used without the authors' permission.

flow, a collection of break points is selected. The break points in turn define document sub-chunks. In an island display, these sub-chunks are displayed in a 2D map as circles. The narrative order is indicated on the map using solid black arrows among the circles.

Figure 4.20 shows the island display of a Buddhist article. The narrative flow of the article is briefly summarized in the display.

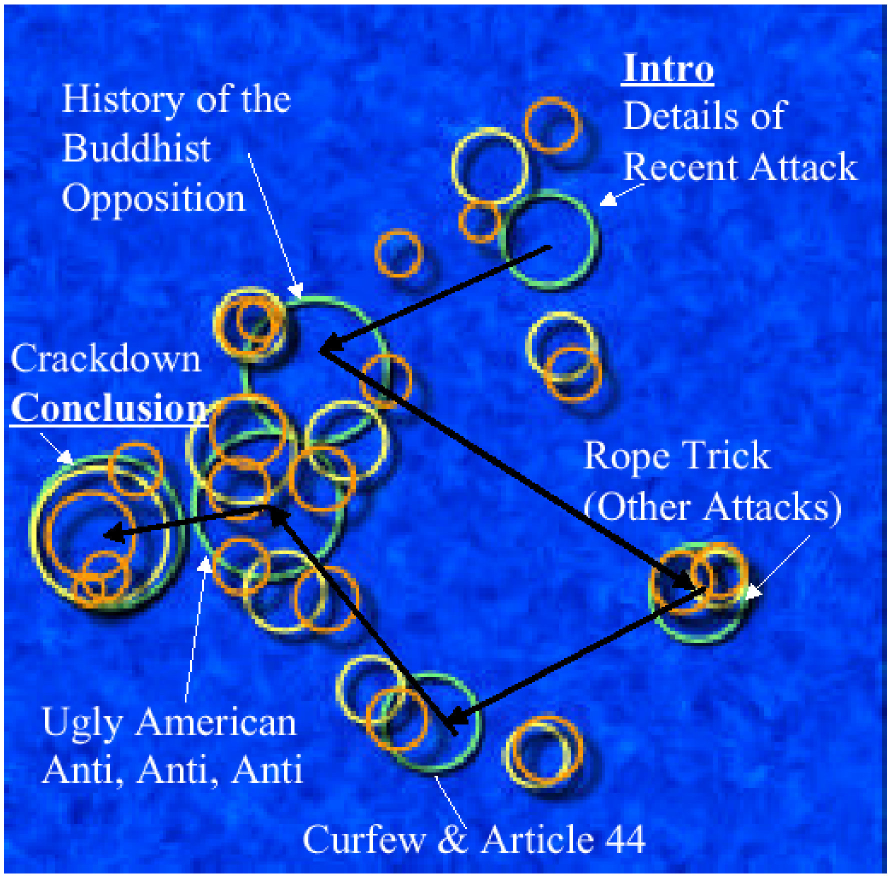


Figure 4.20: The island display of a Buddhist article [MWBF98]. This figure is used without the authors' permission.

4.2 Visualization Space

Definition: *visualization* - the process that generates graphics from the data set automatically. Interaction can be involved in this process.

The visualization space corresponds to the third level of the KRA model, the *description of the memories* level. The visualization approach varies a lot in different applications. I present a list of sub-spaces of the visualization space, i.e., visualization structure space, visual encoding space and screen space, and illustrate how the simplification operators can be applied upon them using examples.

4.2.1 Visualization Structure Space

Definition: *visualization structure* - the organization of a graphics generation process or a visual interaction process. In this space, steps in the process are the objects in the input sets of the simplification operators.

The generalization operators can be used upon the visualization structure space for simplification. For example, [Che03] proposes a conceptual model called compound brushing for modeling the brushing techniques used in dynamic data visualization. Figure 4.21 gives an example of how the structure of a complex brushing operation can be simplified [Che03]. The top window shows a brushing process that is a join of three brushes in three dimensions. The bottom window shows the abstracted view of the structure of the brushing. Users can easily modify the brushing process by dragging and dropping visual entities in the abstracted view of the brushing process.

4.2.2 Visual Encoding Space

Definition: *visual encoding* - the mappings between data attributes and visual attributes. In this space, the mappings are the objects in the input sets of the simplification operators.

An attribute in the data space (*data attribute*) is often mapped to one or several visual attributes (*visual attributes*). I omit the latter case since several visual attributes can be considered as a compound visual attribute. For an accurate visual representation, each distinct value of the data attribute existing in the data set should have a distinct mapped value in the corresponding visual attribute. In practice, it cannot always be satisfied. For example, a color scale of 360 entries cannot satisfy a data attribute with more than 360

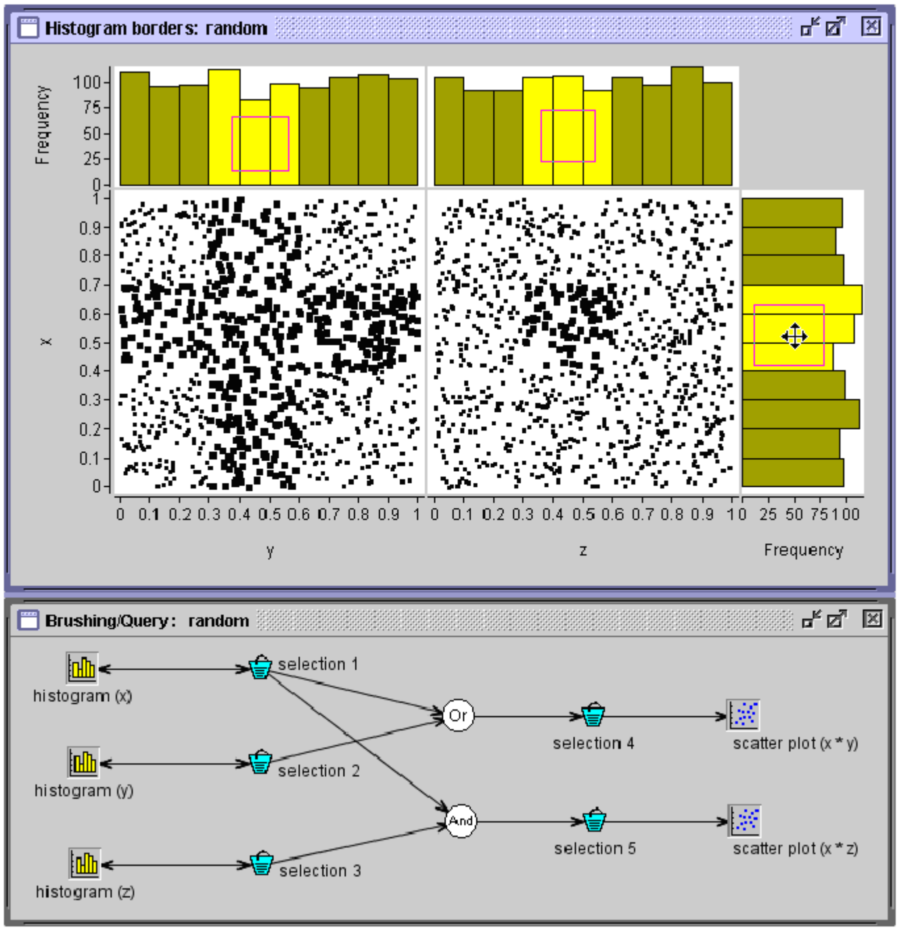


Figure 4.21: The Compound Brush [Che03]. This figure is used without the authors' permission.

distinct values. Thus simplification in the visual encoding space, such as mapping multiple values to the same color (aggregation), is sometimes required rather than preferred.

For example, the images showed in the bottom windows in Figure 4.22 [WY04] convey a real attribute using colors. The top windows in Figure 4.22 show the color mapping of their bottom windows. Aggregation is applied upon the visual encoding space since there are more distinct values in the real attribute than distinct colors in the color scale. In Figure 4.22 (a), aggregation is applied uniformly across the whole color space. Thus the data set is visualized in the same level of detail. While in Figure 4.22 (b), the mappings between the real attribute and the green and blue colors are less aggregated than those of the red and pink colors. Thus a part of the data set is visualized in more detail than the other parts.

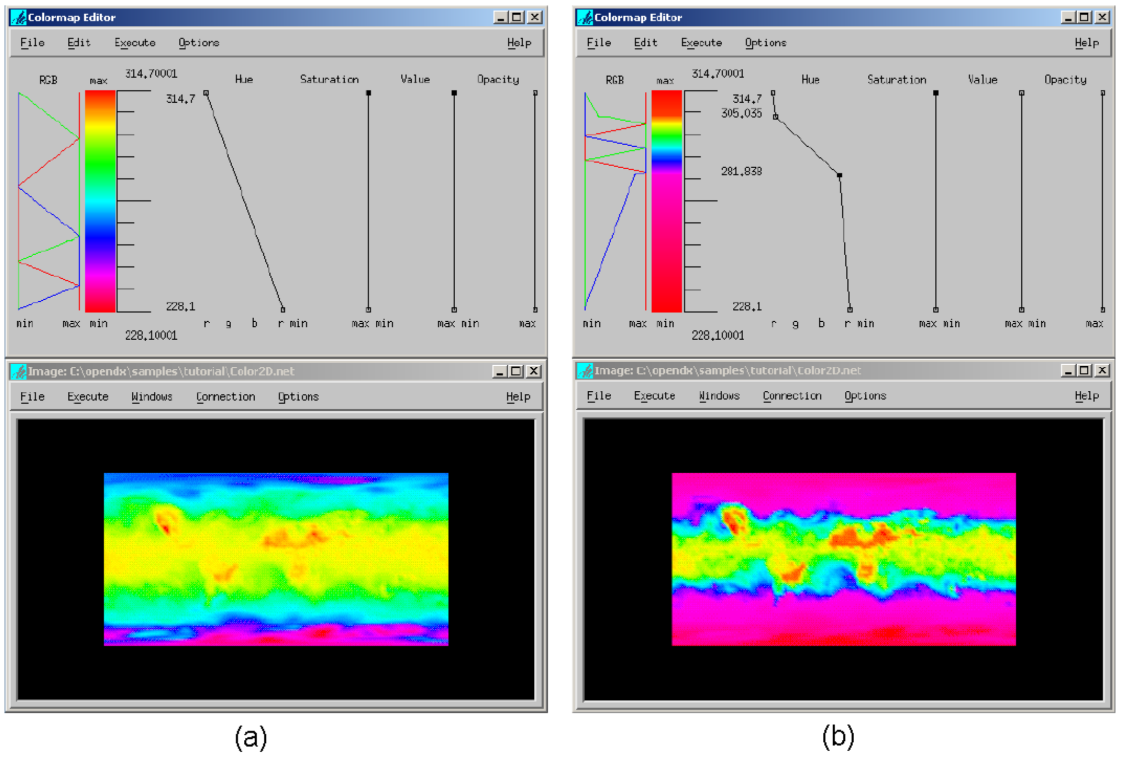

Colormap Editor
File Edit Execute Options Help
RGB max 314.70001 Hue Saturation Value Opacity
314.7
305.035
281.833
228.1
min max min 228.10001 r g b r min max min max min max
Image: C:\opendx\samples\tutorial\Color2D.net
File Execute Windows Connection Options Help
(a)
(b)

Figure 4.22: The mapping between a real attribute and the colors (a) The real value range is uniformly mapped to the colors. (b) The mapping to the green and blue colors is less aggregated than that to the pink and red colors. Images generated with OpenDX.

For another example, Table Lens [RC94] applies an approximate operator upon the visual encoding space for multi-resolution visualization of tables. In traditional tables, each value in the data set is visualized using a string of characters. In a Table Lens display, values out of focus are approximately represented using visual elements that are less expensive than strings (see Figure 4.23 [RC94]). For example, a real value is represented using a line whose length is proportional to the value, while a nominal value is represented by a small block whose color and its positioning within the grid cell indicates that value.

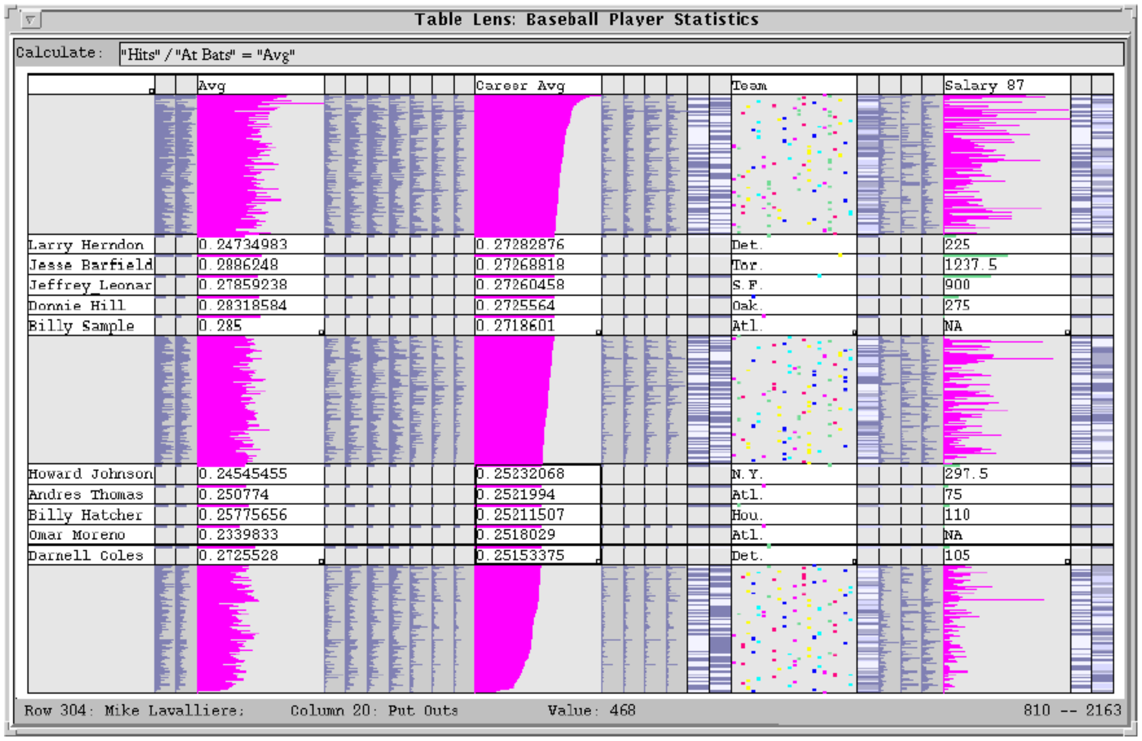


Table Lens: Baseball Player Statistics

Calculate: "Hits" / "At Bats" = "Avg"

	Avg	Career Avg	Team	Salary 87
Larry Herndon	0.24734983	0.27282876	Det.	225
Jesse Barfield	0.2886248	0.27268818	Tor.	1237.5
Jeffrey Leonar	0.27859238	0.27260458	S.F.	900
Donnie Hill	0.28318584	0.2725564	Oak.	275
Billy Sample	0.285	0.2718601	Atl.	NA
Howard Johnson	0.24545455	0.25232068	N.Y.	297.5
Andres Thomas	0.250774	0.2521994	Atl.	75
Billy Hatcher	0.25775656	0.25211507	Hou.	110
Omar Moreno	0.2339833	0.2518029	Atl.	NA
Darnell Coles	0.2725528	0.25153375	Det.	105

Row 304: Mike Lavalliere; Column 20: Put Outs Value: 468 810 -- 2163

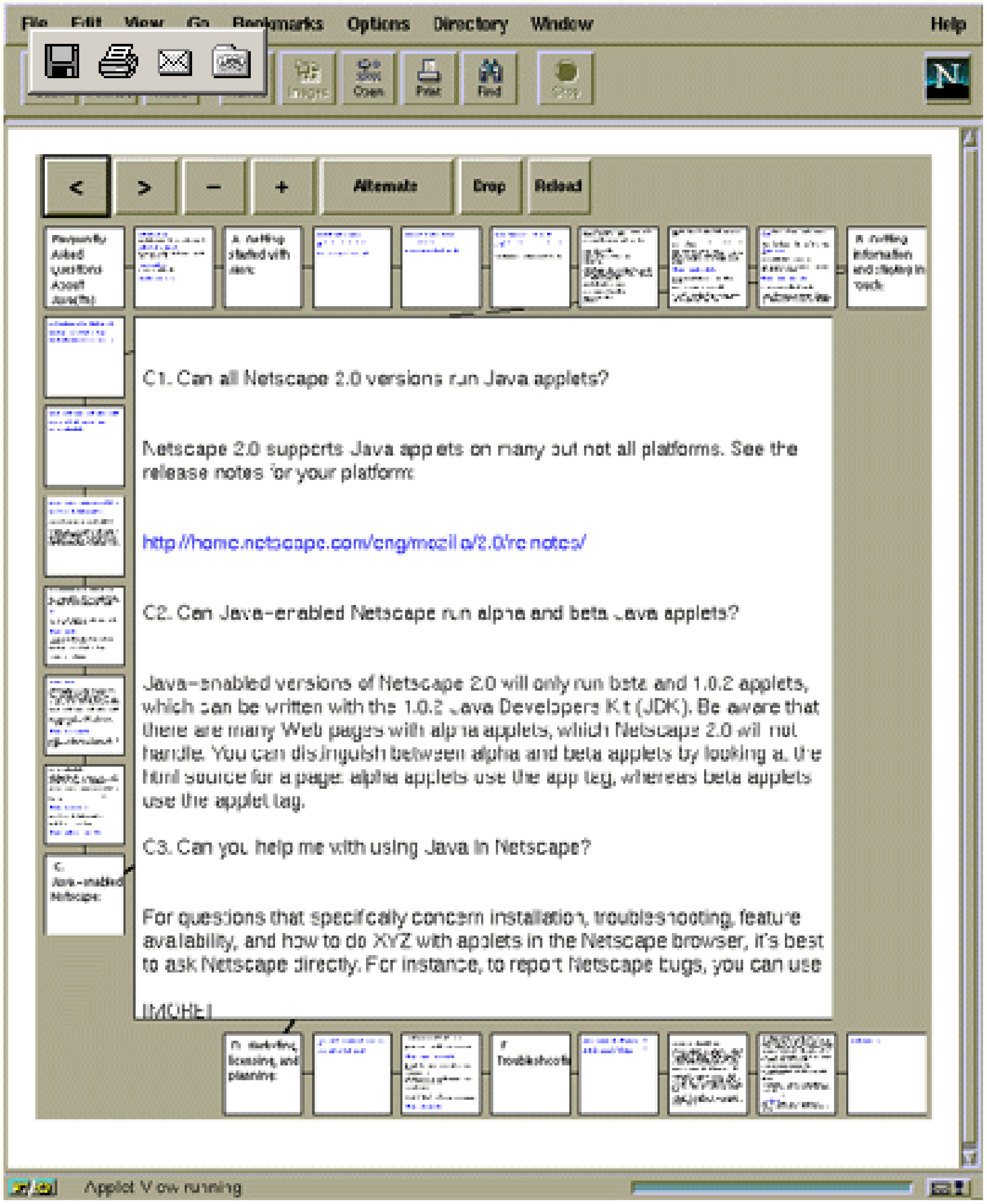
Figure 4.23: A basketball player statistics data set shown in the Table Lens [RC94]. This figure is used without the authors' permission.

4.2.3 Screen Space

Definition: *screen space* - the pixels composing a display. In this space, pixels are the objects in the input sets of the simplification operators.

Simplification in screen space happens when there are fewer pixels than needed for a display. For example, if a display requires 100,000 pixels while there are only 10,000 pixels available on the screen, sampling, aggregation, or approximation needs to be used to fit the display into the available pixels.

A typical example of simplification in screen space is thumbnails. Thumbnails refer to the miniature pictures of large pictures. They are often used in multi-resolution visualization systems. For example, the Zoom Browser [Hol97, BHR99] visualizes non-focus web pages as thumbnails and places them around the focus page to provide context. Figure 4.24 shows a screen dump [Hol97] of the Zoom Browser.



C1. Can all Netscape 2.0 versions run Java applets?

Netscape 2.0 supports Java applets on many but not all platforms. See the release notes for your platform:

http://home.netscape.com/eng/mozilla/2.0/relnotes/

C2. Can Java-enabled Netscape run alpha and beta Java applets?

Java-enabled versions of Netscape 2.0 will only run beta and 1.0.2 applets, which can be written with the 1.0.2 Java Developers Kit (JDK). Be aware that there are many Web pages with alpha applets, which Netscape 2.0 will not handle. You can distinguish between alpha and beta applets by looking at the html source for a page: alpha applets use the app tag, whereas beta applets use the applet tag.

C3. Can you help me with using Java in Netscape?

For questions that specifically concern installation, troubleshooting, feature availability, and how to do XYZ with applets in the Netscape browser, it's best to ask Netscape directly. For instance, to report Netscape bugs, you can use

[MORE]

Figure 4.24: The Zoom Browser displays context web pages as tiles around the focus page [Hol97]. Each tile contains a thumbnail or a summary of a context web page. This figure is used without the authors' permission.

4.3 Discussion

In the above sections, I have discussed simplification in different spaces. It should be pointed out that simplification often happens in multiple spaces at the same time.

Figure 4.25 shows an example of simplification in both the data item space and the dimension space at the same time. Figure 4.25 (a) shows the AAUP data set in parallel coordinates. In Figure 4.25 (b), the data item space of the AAUP data set is simplified using the IHD approach while the dimension space of the AAUP data set is simplified using the VHDF approach.

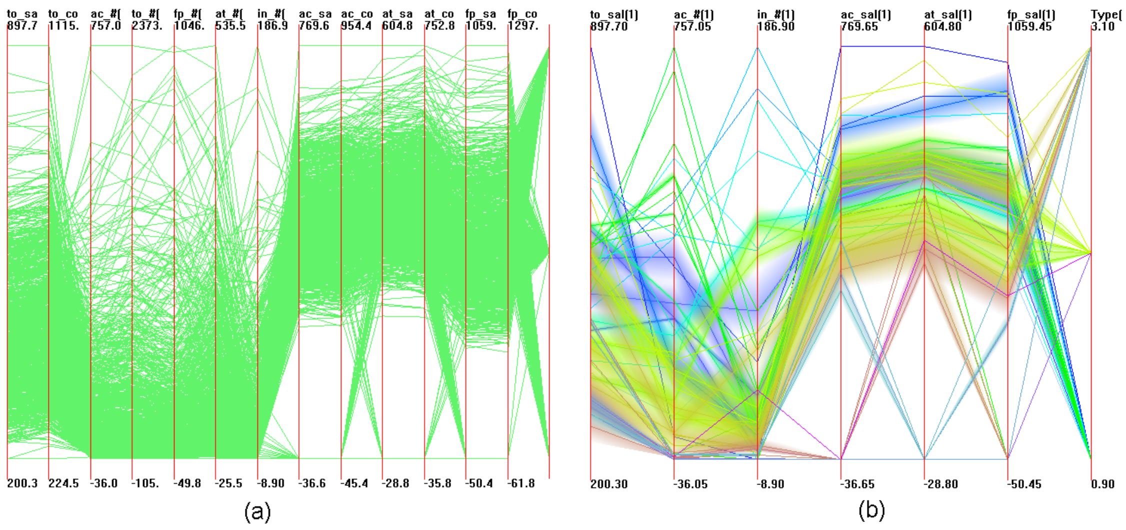


Figure 4.25: The AAUP data set in parallel coordinates. (a) Before simplification (b) The data item space is simplified using the IHD approach while the dimension space is simplified using the VHDF approach.

Combining the simplification operators and the spaces in different ways leads to different multi-resolution visualization approaches. I believe that there are many combinations that have never been tried. New multi-resolution visualizations can be generated using such combinations. However, their effectiveness needs to be formally evaluated.

Chapter 5

Interactive Visualization

Having the views at multiple levels of detail, a multi-resolution visualization system needs to visually convey the views to users and allow them to interactively navigate among the views. In my original plan, I had intended to write two chapters with regard to this topic. One chapter to describe the general approaches to presenting multiple views, such as the zoomable interface, the overview + detail interface, and the focus + context interface, and the other chapter to describe the general interaction tools, such as zooming, and panning, selection, distortion, and overlap reduction. However, I found that it is hard to write the view presentation approaches separately from the interaction tools. For example, the zoomable interface is closely related to zooming and panning techniques, the overview + detail interface is closely related to selection techniques, and the focus +context interface is entangled with distortion techniques. Thus I decided to use a mixed approach that presents the view presentation and interaction tools together.

In Section 5.1 I present the zoomable interface and zooming and panning operations. In Section 5.2 I introduce the overview + detail interface and the selection operation. In Section 5.3 I describe the focus + context interface and the distortion and overlap deduction operations. In Section 5.4 I talk about other common user interactions in multi-resolution visualization systems.

Please note that an operation discussed together with a certain user interface means that the operation is frequently used in the interface rather than that the operation can only be used in the interface.

5.1 Zoomable Interfaces and Zooming and Panning Operations

Definition: *zoomable interface* - an interface in which users navigate among views using zooming operations and navigate within the same view using panning operations.

Definition: *zooming in* - the interaction that changes the current display from a view of a lower level of detail to a view of a higher level of detail.

Definition: *zooming out* - the interaction that changes the current display from a view of a higher level of detail to a view of a lower level of detail.

Definition: *panning* - the interaction that changes the current display from a sub-region of a view to an adjacent sub-region of the same view. There can be overlaps between the two regions.

5.1.1 Zoomable Interfaces

The interactive map visualization provided by www.mapquest.com uses a typical zoomable interface. Figure 5.1 shows an example of the interactive map. The target address is Worcester Polytechnic Institute (WPI), the university of the author. It is marked by a red star in the maps. Figure 5.1 (a) shows a map in a low level of detail where only states can be seen. After the user clicks a zoom button representing a higher level of detail, a map in a higher level of detail (Figure 5.1 (b)) is shown. In this map towns of Worcester can be seen. After the user clicks another zoom button, a more detailed map (Figure 5.1 (c)) is shown. In this map the street location of WPI is revealed. Then the user clicks a neighbor block of WPI. The focus of the map switches from WPI to the clicked block. During the visual exploration, the user performed two zooming operations to get Figure 5.1 (b) and (c), and a panning operation to get Figure 5.1 (d).

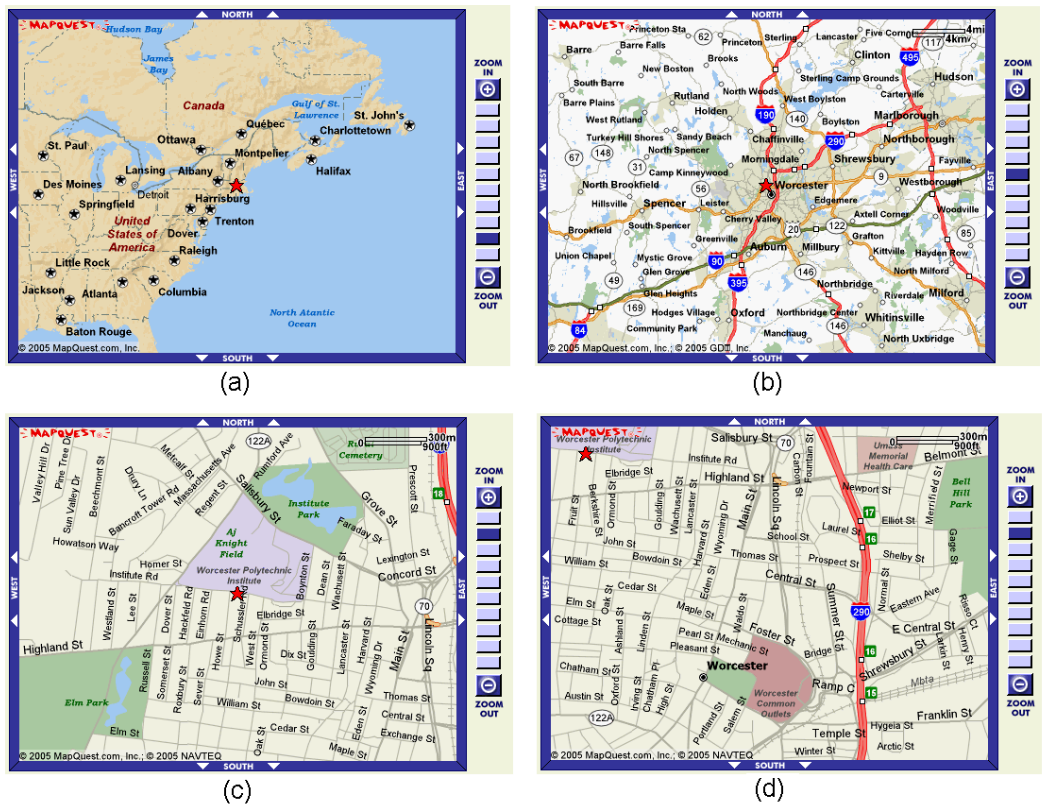

NORTH
SOUTH
WEST
EAST
ZOOM IN
ZOOM OUT
MAPQUEST
© 2005 MapQuest.com, Inc.
© 2005 MapQuest.com, Inc.; © 2005 GDT, Inc.
© 2005 MapQuest.com, Inc.; © 2005 NAVTEQ
Worcester Polytechnic Institute
Worcester
(a)
(b)
(c)
(d)

Figure 5.1: The map of WPI. (a) The map in a low level of detail. (b) (a) after zooming in. (c) (b) after zooming in. (d) (c) after panning. These figures were captured from www.mapquest.com.

5.1.2 Zooming and Panning

[HBP02] states that panning and zooming can be linear or nonlinear. When panning and zooming are controlled with the mouse or the keyboard, a change in the input device is usually linearly related to how much is panned or zoomed. Nonlinear panning and zooming have been proposed in three forms: (a) goal-directed zoom, where direct zooming to an appropriate scale is supported [WLS98]; (b) combined zooming and panning, where extensive panning automatically leads to zooming [IH00]; and (c) automatic zoom to objects, where a click with the mouse on an object automatically zooms to center on that object [FZ98, War00].

5.2 Overview + Detail Interfaces and Selection

Definition: *overview + detail interface*: an interface composed of multiple windows. Some windows provide context and easy navigation (overview windows) for other windows (detail windows).

Definition: *selection* - the interaction that isolates a subset of entities on a display for further operations [Wil96].

5.2.1 Overview + Detail Interfaces

There are several types of overview + detail interfaces. In the first type, an overview window visually presents views of a lower level of detail than those presented in the detail windows. Thus the overview window shows a large region of the data set than the detail windows and provides "context" for the detail windows. When users want to change the sub-region displayed in a detail window, they can select a sub-region of interest from the overview window and the selected sub-region will be displayed in the detail window. Compared to panning in the detail window, selecting a sub-region allows users to jump from one sub-region to a non-adjacent one.

Figure 5.2 [NSP96] is an example of the first type of overview + detail interface. It shows the interface of the Visible Human Explorer (VHE) visualization system [NSP96] that is used to remotely explore the Visible Human digital library and retrieve images. In the figure, the left view is the overview window. It displays a miniature coronal section, a front-view longitudinal cut of the body. It acts as an overview of the library, giving the users a general understanding of the contents of the body from head to toe. The right top view is the detail window. It displays a miniature axial section, the type of cut typically seen in medical atlas textbooks. A horizontal indicator on the overview indicates the vertical position, on the body, of the axial section shown in the detail window. The indicator is attached to a vertical slider widget spanning the height of the overview and can be dragged vertically across the body to sweep the cut, shown in the detail window, through the body. Thus the slider allows users to perform selection in the overview window.

In the second type of overview + detail interface, an overview window not only visually conveys available sub-regions, but also conveys available views of different levels of detail as visual entities. Users are not only able to select sub-regions of interest, but also select desired levels of detail for the views displayed in the detail window. The Interactive Hierarchical Displays (IHD) approach [YWR03b] uses such an interface. IHD constructs a large data set into a data hierarchy and allows users to interactively select data clusters displayed. In the overview window, the whole data hierarchy is displayed in a compact triangle. Users can select a region of interest and desired levels of detail from the overview window. The selection is performed through mouse clicking on the overview window or keyboard input. In the detail window, the clusters in the selected region are displayed at selected levels of detail.

Figure 5.3 (a) shows the overview and detail windows of the IHD approach. In the overview window, the contours of clusters in the selected region and at selected levels of detail are highlighted by colors. In the detailed window, these clusters are displayed using the hierarchical parallel coordinates display. Figure 5.3 (b) shows how the detail window changed after the user selected another level of detail in the overview window.

In the third type of overview + detail interface, it is hard to distinguish which win-

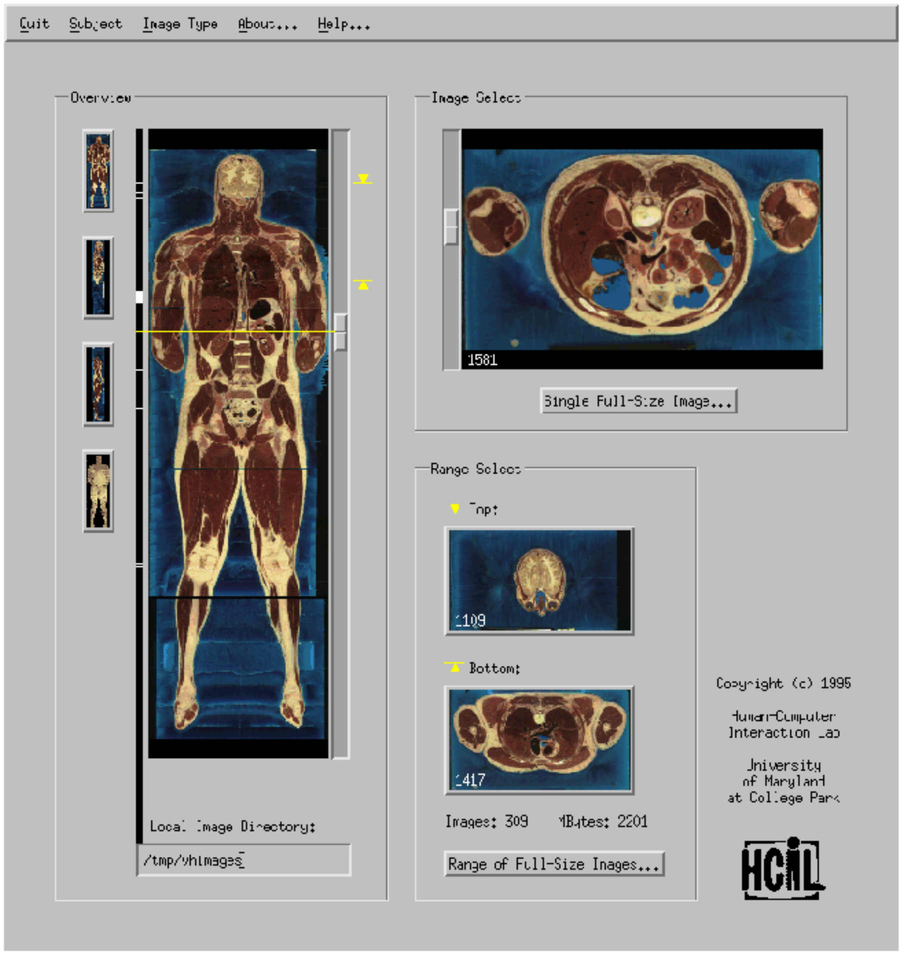
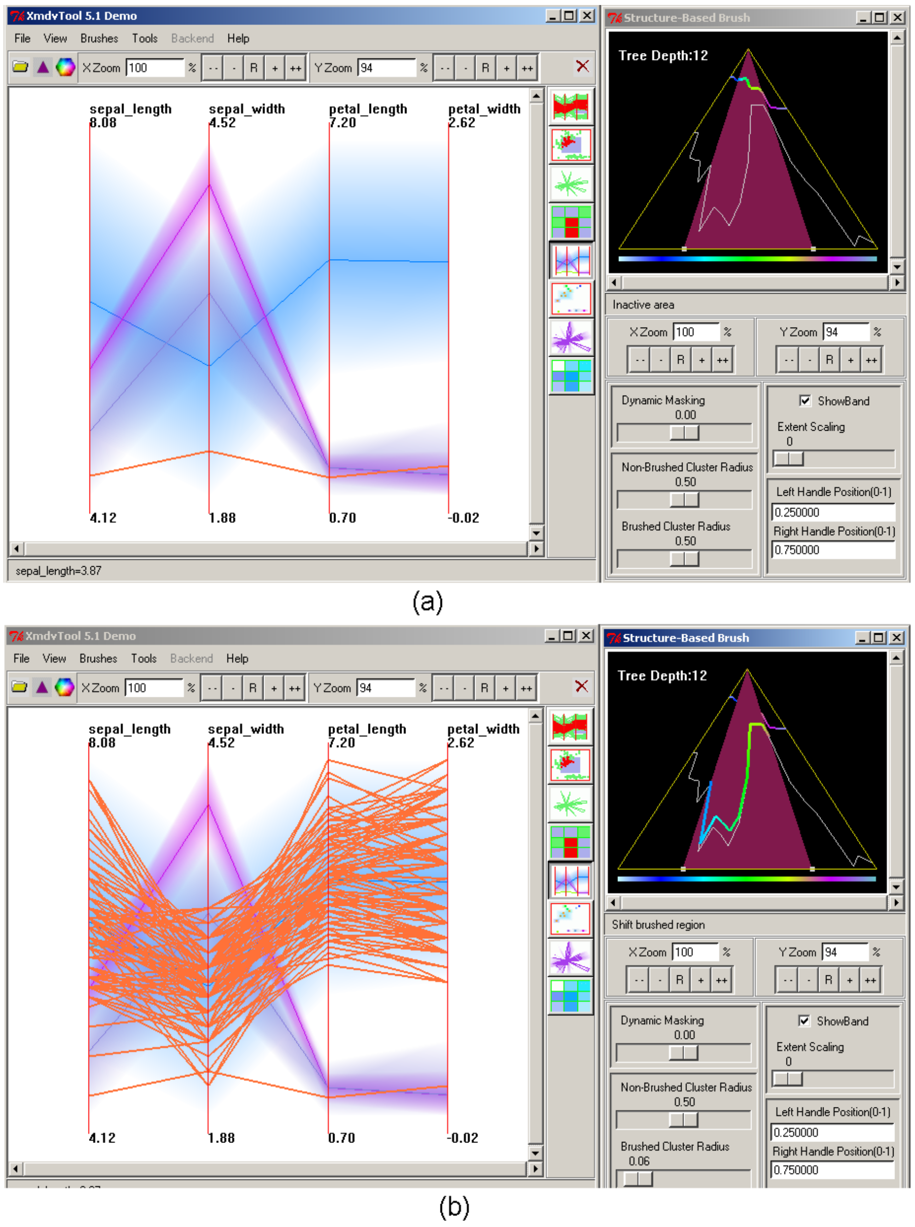

Quit Subject Image Type About... Help...
Overview
Image Select
1581
Single Full-Size Image...
Range Select
Top:
1109
Bottom:
1417
Images: 309 MBytes: 2201
Range of Full-Size Images...
Local Image Directory:
/tmp/vhimages
Copyright (c) 1995
Human-Computer Interaction Lab
University of Maryland at College Park
HCIL

Figure 5.2: The Visible Human Explorer user interface [NSP96], showing a reconstructed coronal section overview and an axial preview image of the upper abdominal region. Dragging sliders animates the cross-sections through the body. This figure is used without the authors' permission.

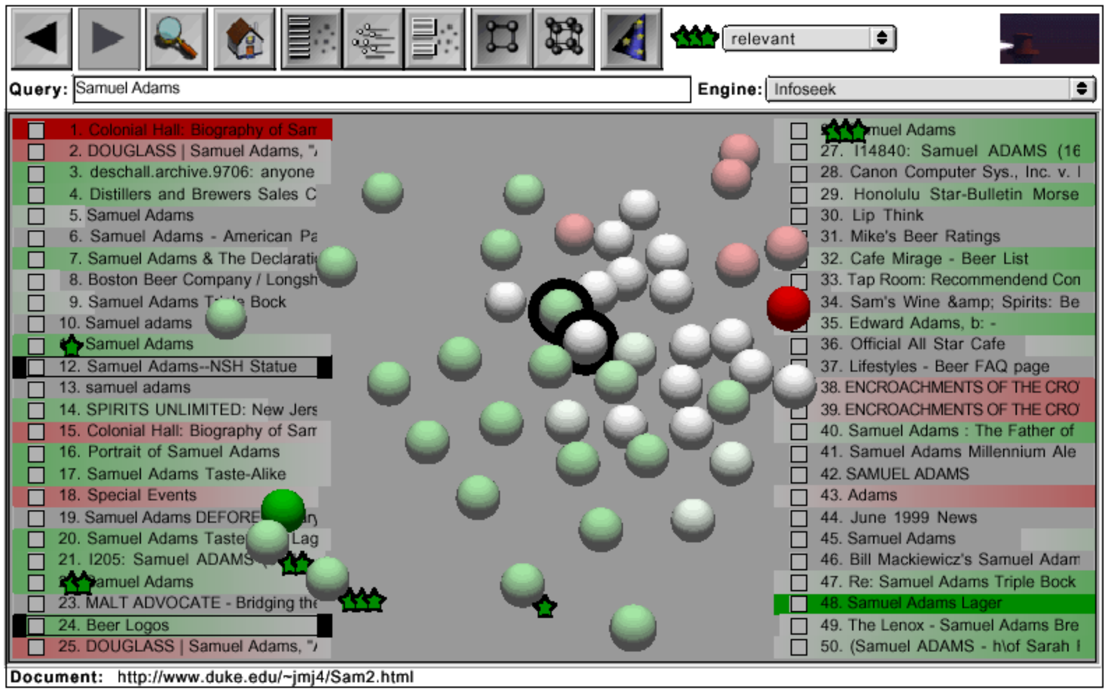
dows are the overview windows and which are the detail windows. For example, the Lighthouse system [LA00] uses two windows (they are mixed together, as seen in Figure 5.4) to display documents retrieved by a document search engine for user queries. In the middle window, each document is generalized into a sphere. The similarities among the documents are revealed by the positions of the spheres in a 2D or 3D space. In the side windows, the documents are displayed in more detail - the names of the documents are displayed. However, the relationships among the documents are revealed in less detail



XZoom 100 %
YZoom 94 %
sepal_length 8.08 4.12
sepal_width 4.52 1.88
petal_length 7.20 0.70
petal_width 2.62 -0.02
sepal_length=3.87
Inactive area
Shift brushed region
Dynamic Masking 0.00
ShowBand
Extent Scaling 0
Non-Brushed Cluster Radius 0.50
Brushed Cluster Radius 0.50
Brushed Cluster Radius 0.06
Left Handle Position(0-1) 0.250000
Right Handle Position(0-1) 0.750000

Figure 5.3: The Interactive Hierarchical Displays Interface. (a) The left window is the detail window while the right window is the overview window. Users control the displayed region and levels of detail in the detail window through the overview window. (b) The detail window changed after the user selected another level of detail in the overview window.

than in the middle window. Users can change their focus in both the middle window and the side windows. It is hard to decide which one is the overview window and which one is the detail window.


relevant

Query: Samuel Adams

Engine: Infoseek

1. Colonial Hall: Biography of Sam
2. DOUGLASS | Samuel Adams, "
3. deschall.archive.9706: anyone
4. Distillers and Brewers Sales C
5. Samuel Adams
6. Samuel Adams - American Pa
7. Samuel Adams & The Declarati
8. Boston Beer Company / Longsh
9. Samuel Adams T---e Bock
10. Samuel adams
Samuel Adams
12. Samuel Adams--NSH Statue
13. samuel adams
14. SPIRITS UNLIMITED: New Jers
15. Colonial Hall: Biography of Sam
16. Portrait of Samuel Adams
17. Samuel Adams Taste-Alike
18. Special Events
19. Samuel Adams DEFORE
20. Samuel Adams Taste Lag
21. I205: Samuel ADAMS
amuel Adams
23. MALT ADVOCATE - Bridging the
24. Beer Logos
25. DOUGLASS | Samuel Adams, "

muel Adams
27. I14840: Samuel ADAMS (16
28. Canon Computer Sys., Inc. v. I
29. Honolulu Star-Bulletin Morse
30. Lip Think
31. Mike's Beer Ratings
32. Cafe Mirage - Beer List
33. Tap Room: Recommendend Con
34. Sam's Wine &amp; Spirits: Be
35. Edward Adams, b: -
36. Official All Star Cafe
37. Lifestyles - Beer FAQ page
38. ENCROACHMENTS OF THE CRO
39. ENCROACHMENTS OF THE CRO
40. Samuel Adams : The Father of
41. Samuel Adams Millennium Ale
42. SAMUEL ADAMS
43. Adams
44. June 1999 News
45. Samuel Adams
46. Bill Mackiewicz's Samuel Adam
47. Re: Samuel Adams Triple Bock
48. Samuel Adams Lager
49. The Lenox - Samuel Adams Bre
50. (Samuel ADAMS - h\of Sarah I

Document: http://www.duke.edu/~jmj4/Sam2.html

Figure 5.4: A screen shot of the Lighthouse system [LA00]. In the middle of the display, documents are visualized as floating spheres. In the sides of the display, the same documents are visualized using rank list. This figure is used without the authors' permission.

In the fourth type of overview + detail interface, the overviews and detailed views are combined into the same window, but they can still be clearly distinguished through the visualization structure or visual mapping. This is different from the focus + context interface. Figure 5.5 shows an overview + detail interface of the fourth type. In this display, the overview is mapped to the position attribute while the detailed view is mapped to the shape attribute. In the display, the Cars data set (7 dimensions) is visually represented. Each data item in the data set is mapped to a star glyph. 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. Thus the glyph shapes provide a detailed view of the data set to users. At the same time, the star glyphs are placed in a 2D space whose x- and y-axes are the first two principal components of the data sets to form a scatterplot display. If only the positions of the star glyphs are considered, it is a view that abstracts the data set from a 7 dimensional space to a 2 dimensional space. It provides an overview of the similarity among the data items. Thus an abstracted view and a detailed view are combined into the same display, but distinguished by different visual attributes.

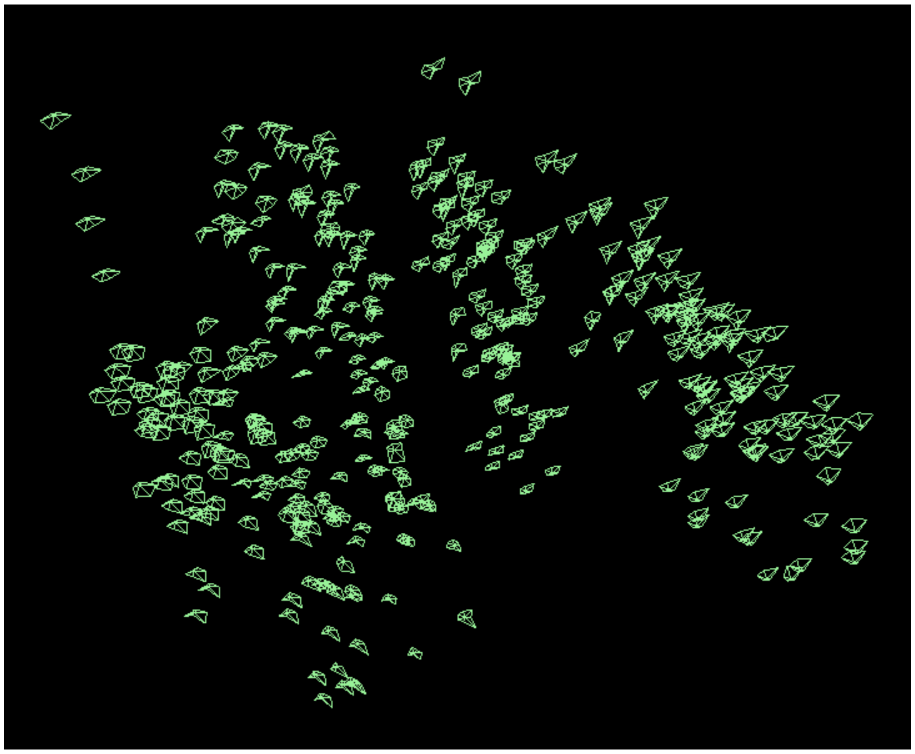


Figure 5.5: A star glyph display of a 7 dimensional data set. The shapes of the glyphs convey detail of all the 7 dimensions, while the positions of the glyphs convey a projection of the data set in a 2 dimensional space.

Figure 5.6 [XMD] shows another example of such an interface. It displays a scatterplot matrix of the Cars data set that contains 7 dimensions. In the display, each pair of dimensions generates a scatterplot. Each data item is projected to these plots. Except for the diagonal plots (from top left to bottom right), the position of the projected point in a plot is decided by the values of the data item in the two dimensions that correspond to this plot. The diagonal plots are used to display one-dimensional histograms of the diagonal dimensions. Thus, the diagonal plots provide overviews while the other plots provide detailed views for the data set.

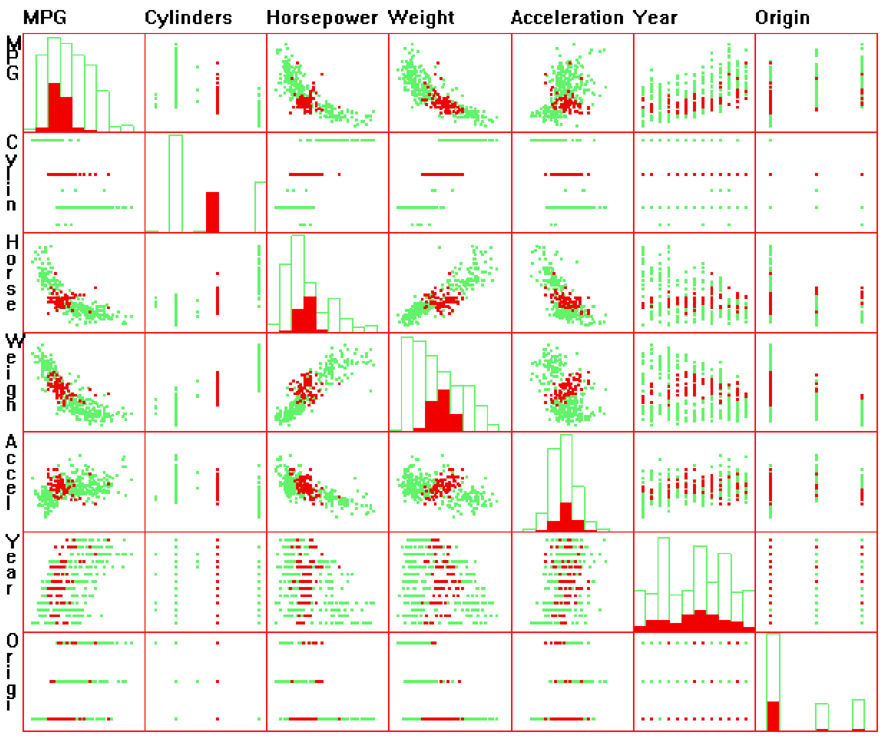


Figure 5.6: A scatterplot matrix display of the Cars data set. One dimensional histograms of the dimensions are displayed in their corresponding diagonal plots and provide overviews of the dimensions. The projections and aggregations of brushed data items are highlighted in red.

Figure 5.7 shows a similar approach to the above example. In this figure [HLD02], 1-D histograms are added into a parallel coordinates display of the Cars data set. Each dimension in the data set is represented as a uniformly spaced vertical axis. Each data item in this multi-dimensional space is mapped to a polyline that traverses across all the axes and composes a detailed view. A one-dimensional histogram overlays its corresponding axis and provides an overview of that dimension.

5.2.2 Discussion

Using an overview + detail interface, the users changes the display in the detail window by performing selections in the overview window. According to different interface design, a

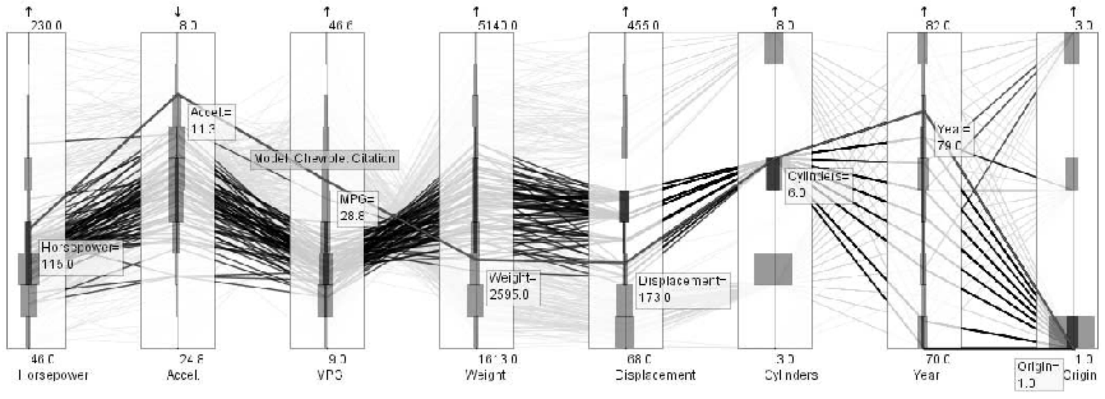


Figure 5.7: A parallel coordinates display of the Cars data set. 1 dimensional histograms overlay their corresponding axes and provide overviews of the dimensions [HLD02]. This figure is used without the authors' permission.

selection in the overview window can cause an immediately change in the detail window, or delayed until the users trigger the refresh of the detail window.

[HBP02] summarizes the benefits of using overview + detail interfaces:

- Navigation is more efficient because users may navigate using the overview window rather than the detail window [BW90]

- The overview window aids users in keeping track of their current position in the information space [PCS95]. The overview window itself might also give users task-relevant information, for example, by enabling users to read section titles from an overview of a document [HF01].

- The overview gives users a feeling of control [Shn97].

[HBP02] lists a number of empirical studies showing that having an overview improves user satisfaction and efficiency over detail-only interfaces without an overview. For example, North and Shneiderman [NS00] compared 18 subjects' performance with a detail-only, an uncoordinated overview + detail, and a coordinated overview + detail interface for browsing textual population data. Compared to the detail-only interface, the coordinated interface was 30-80% faster and scored significantly higher on a satisfaction questionnaire.

[HBP02] also summarizes drawbacks of using overview + detail interfaces:

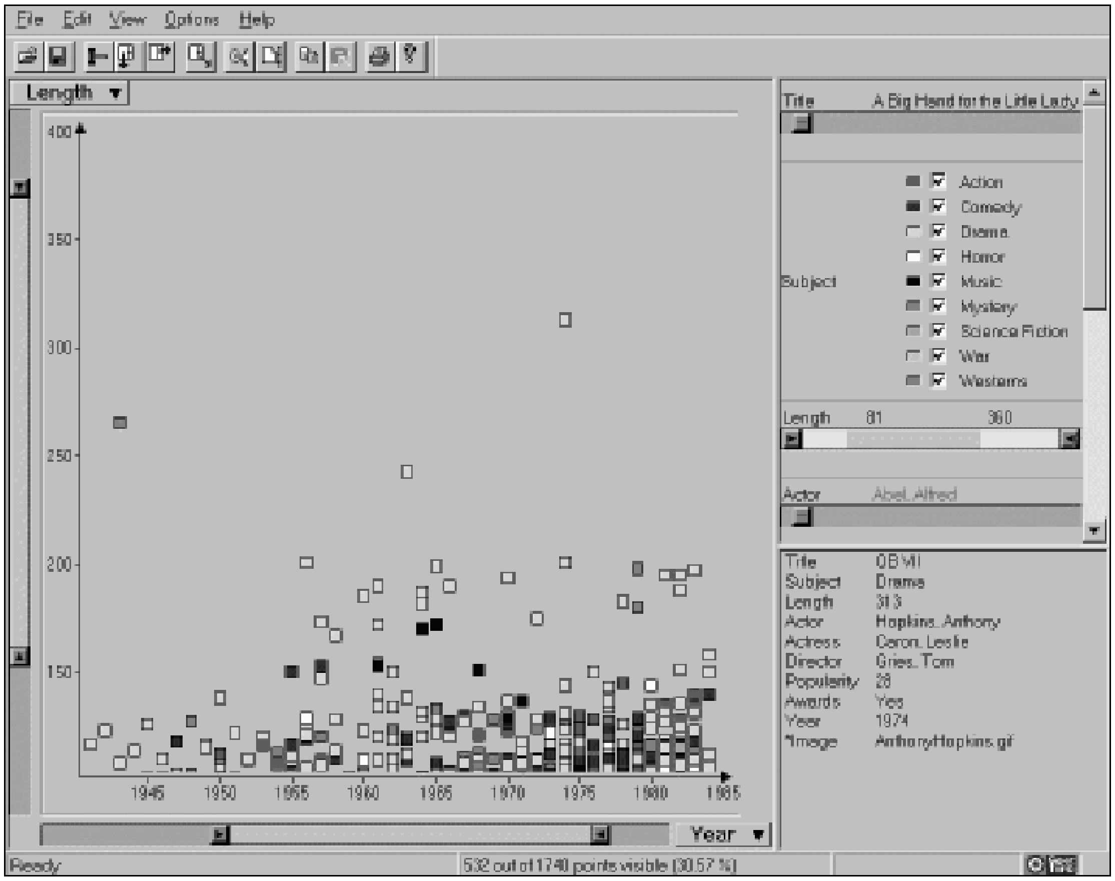
- The spatially indirect relation between overview and detail windows might strain memory and increase the time used for visual search [CMS99].
- Overview + detail interfaces require more screen space than interfaces without overviews.

5.2.3 Selection

Selection is a process whereby a subset of entities on a display is isolated for further operations, such as highlighting, deleting, or analysis [FWR99a]. Wills [Wil96] defined a taxonomy of selection operations, classifying techniques based on whether memory of previous selections is maintained or not, whether the selection is controlled by the underlying data or not, and what specific interactive tool (e.g., brushing, lassoing) is used to differentiate an area of the display. He also created a selection calculus that enumerates all possible combinations of actions between a previous selection and a new selection (replace, add, subtract, intersect, and toggle) and attempted to identify configurations of these actions that would be most useful.

In the context of the overview + context interface, the major goal of selection is to select views of interest and/or sub-regions of the data set so that the selected sub-regions can be displayed in the detail window in the desired views. This task can be trivial or non-trivial. For example, if the detail and overview windows show a map in different levels of detail, users only need to click the mouse within the overview window to define a new center of the zoomed area in the detail window, or to drag a rectangle to define a new zoomed area in the detail window.

Dynamic query is a more complex selection approach. It is a mechanism for specifying queries and visualizing their results [WS92, Shn94]. Figure 5.8 [TBS97] shows an example of dynamic query. In this example, users define a range in a multi-dimensional space using the widgets along the left, bottom, and right of the screen. Data items in the data set falling into this range are visualized in a scatterplot in the large square drawing area.



Title	A Big Hand for the Little Lady
Subject	▨ ✓ Action
	■ ✓ Comedy
	□ ✓ Drama
	□ ✓ Horror
	■ ✓ Music
	▨ ✓ Mystery
	▨ ✓ Science Fiction
	□ ✓ War
	▨ ✓ Westerns
Length	81 360
Actor	Abel, Alfred

Title	QB VII
Subject	Drama
Length	313
Actor	Hopkins, Anthony
Actress	Caron, Leslie
Director	Gries, Tom
Popularity	28
Awards	Yes
Year	1974
*Image	AnthonyHopkins.gif

532 out of 1740 points visible (30.57 %)

Figure 5.8: A screen dump of Spotfire, which is a sample dynamic query interface, captured by [TBS97]. This figure is used without the authors' permission.

Multi-dimensional brushing in XmdvTool [XMD] is another non-trivial selection tool, which is shown in Figure 5.9. The brush is displayed as a multi-dimensional hyper-box (the blue region) in the display. The boundary of the hyper-box defines a range. Data items falling into this range are selected and highlighted. Users can interactively change the boundary of the hyper-box to change their selection.

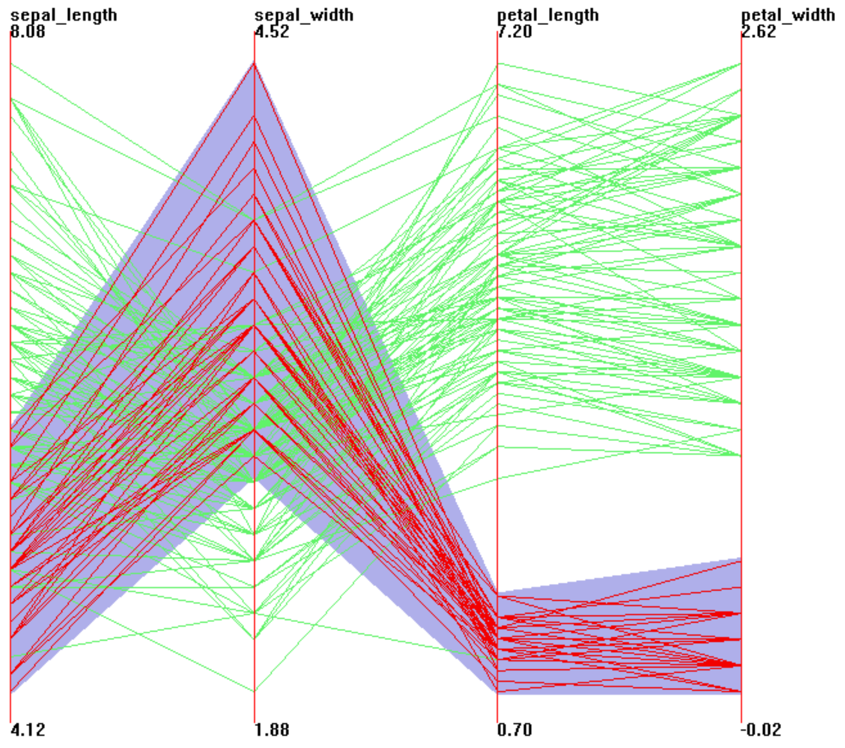


Figure 5.9: Multi-dimensional brushing in XmdvTool. The blue hyper-box is the brush. Data items selected by the brush are highlighted in red.

In the Visual Hierarchical Dimension Reduction (VHDR) approach, users need to select multiple dimension clusters from a hierarchy in the overview window to form a subspace displayed in the detail window. The clusters of interest may be spread throughout the hierarchy. Users can perform such selection by multiple mouse clicking, but they may need to click the mouse many times if users are interested in many dimension clusters. Thus the selection is non-trivial.

In the case of non-trivial selection, automatic tools that help users improve the efficiency of the operation are desired. Such tools perform the non-trivial selection automatically for users according to their intentions.

For example, the VHDR approach [YWRH03] used a selection operation provide by the hierarchy display (InterRing [YWR02, YWRP03]), which is named structure-base brushing, to help users select multiple clusters at one time. Structure-based brushing was first proposed in [FWR99a] and was adapted in InterRing [YWR02, YWRP03]. It allows users to select multiple nodes of a hierarchy at the same time according to certain criteria. For example, the criteria can be the maximum number of leaf node allowed for the nodes selected. When users click the right button of the mouse on a cluster of the InterRing display, a small dialog pops up. Users then select a threshold between 1 and the number of leaf nodes contained in the cluster to trigger the selection. The cluster and all its descendants can be considered as a tree rooted at the cluster. The selection starts from the root cluster. If its number of leaf nodes is less than or equal to the threshold, it is selected and the process stops. Otherwise this process is repeated iteratively for all its children. Using this brush, users can select approximately uniformly sized clusters and cover all the leaf nodes of the root cluster (a leaf node as covered if either itself or one of its ancestors is selected) at one time. For example, in the VHDR approach, if a user wants to select all the original dimensions from the dimension hierarchy, he/she can perform a structure-based brushing on the root node with a threshold of 1.

Figure 5.10 gives an example of a series of structure-based brushing operations. Figure 5.10 (a) shows that structure-based brushing is applied to the root node and the threshold is set to 6 . Figure 5.10 (b) shows the result of this selection. Figure 5.10 (b) shows structure-based brushing is applied again on a region where a lower level of detail is needed. Figure 5.10 (d) shows the selection result. Part of the hierarchy is selected at a lower level of detail than the other parts of the hierarchy.

Some other examples of automatic selection can be seen in the Value and Relation (VaR) display [YPH$^+$04]. For example, the automatic selection tool for separated dimensions of the VaR display takes a user-assigned dimension and correlation threshold as

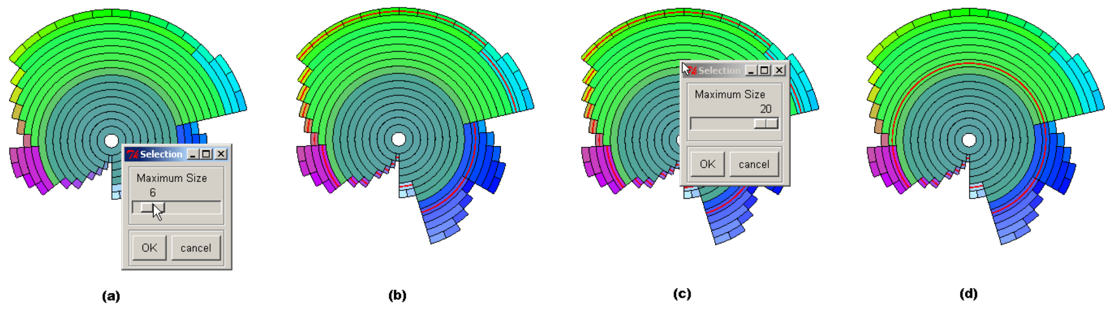


Figure 5.10: Structure-Based Brushing [YWR02]. (a) shows applying structure-based brushing on the root node and setting the threshold to 6. (b) shows the selection result. (c) shows applying structure-based brushing again on a node where a lower level of detail is needed. (d) shows the selection result of these two brushing operations.

input and returns a set of typical dimensions that describe the major features of the data set. The assigned dimension will be included in the returned set of dimensions. Between each pair of dimensions in the result set, the correlation measure is larger than the threshold. For any dimension that is not in the result set, there is at least one dimension in the result set such that the correlation measure between it and the unselected dimension is smaller than the threshold. Using this tool, a user is able to select a set of dimensions to construct a lower dimensional subspace revealing the major features of the data set without much redundancy. Figure 5.11 shows an example of automatic selection for separated dimensions in a high dimensional data set.

The following algorithm is used for the above automatic selection approach:

- Step1: Set the assigned dimension as “selected” and all other dimensions as “unselected”.

- Step2: Find all unselected dimensions whose distances to all existing selected dimensions are larger than the threshold. Mark them as “candidate”.

- Step3: If there is no candidate dimension, go to Step4. Else, set one candidate dimension as “selected” and other candidate dimensions as “unselected”. Go back to Step2.

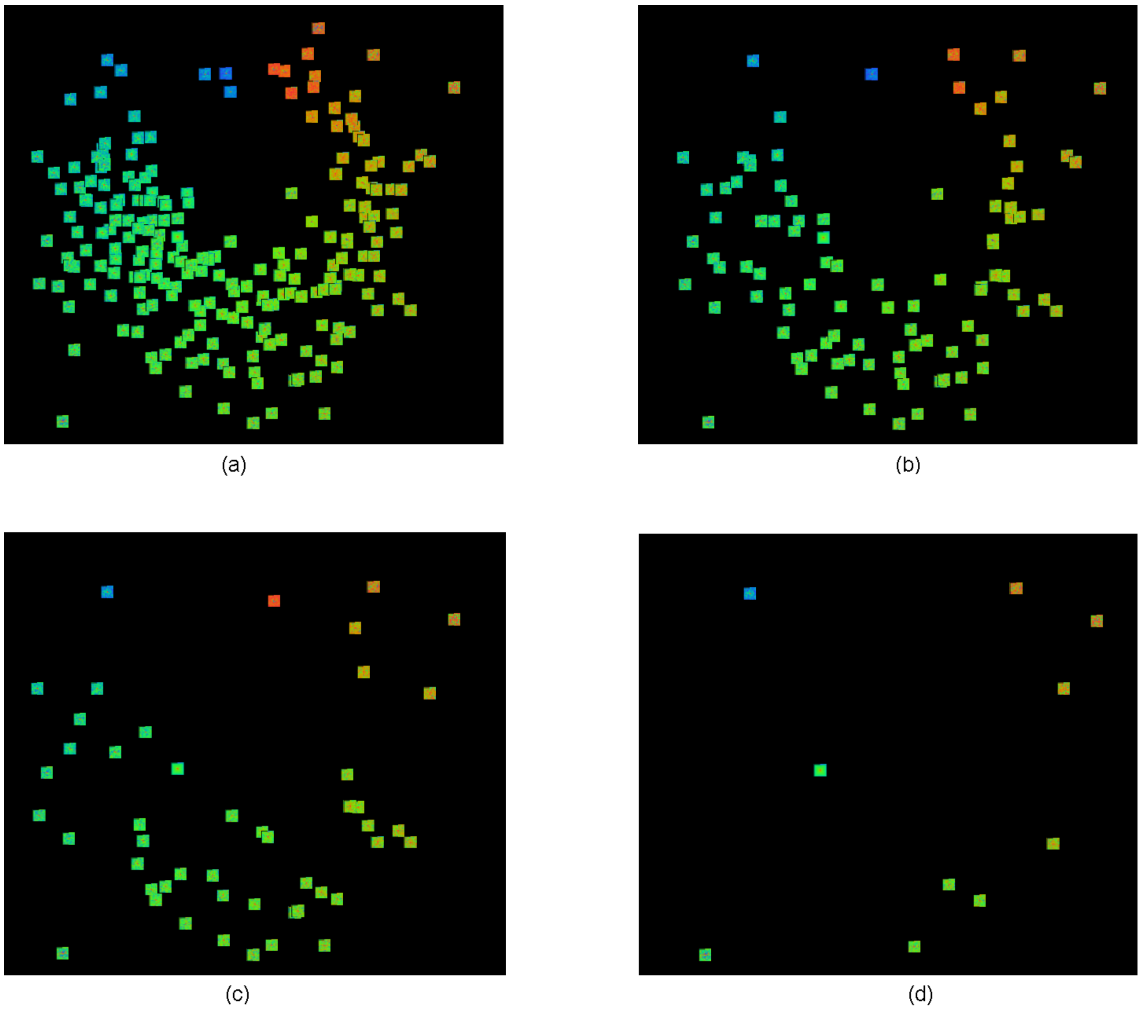


Figure 5.11: Automatic Selection of Separated Dimensions [YPH+04]. Unselected dimensions are hidden using Dynamic Scaling. Selected dimensions in (a)(b)(c)(d) are generated using the automatic selection with the same assigned dimensions and an increasing correlation threshold. The data set is the OHSUMED data set (215 dimensions, 298 data items), which contains the word-counts of a medical abstract collection [HBLH94].

- Step4: Return all dimensions marked as "selected".

5.3 Focus + Context Interfaces and Distortion

Definition: *focus + context interface* - an interface where views of different levels of detail are mixed together in the same display.

Definition: *distortion* - an operation that increase the screen space allocated to some objects in the display while decreasing the screen space allocated to other objects.

5.3.1 Focus + Context Interfaces

A focus + context interface presents views of different levels of detail in the same display. The objects from views of lower levels of detail provide a context for the objects from views of higher levels of detail (focus). An interactive multi-resolution visualization system usually allows users to interactively set or change focus and context through distortion.

For example, Figure 5.12 [WY04] presents a scatterplot matrix where a fisheye lens [Fur86] is used to increase the screen space allocated to some scatter plots by decreasing the screen space allocated to their adjacent plots.

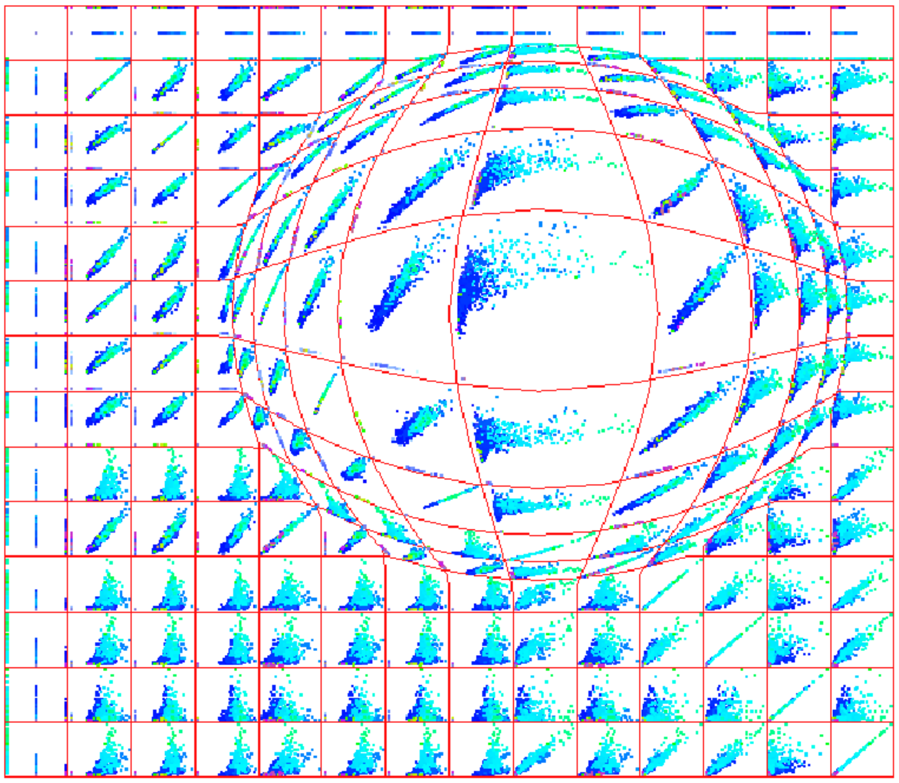


Figure 5.12: A focus + context scatterplot matrix display [WY04].

For another example, Figure 5.13 shows a focus + context interface in a hierarchy visualization system [YWR02]. Multi-focus distortion is performed to set two foci in the display through mouse clicking and drag-and-drops. The levels of detail of the clusters in the focus region are raised in the cost of lowering the levels of detail of their adjacent clusters. In this case, the distortion happens in the angles assigned to each cluster. While in the focus + context display shown in Figure 5.14, the distortion happens in the radii of the cluster layers.

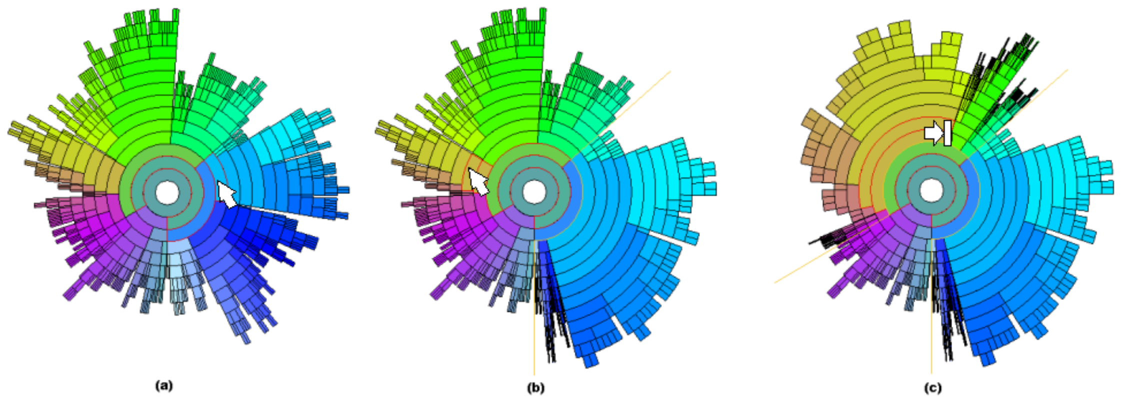


Figure 5.13: Multi-Focus Distortion [YWR02]. (a) shows the hierarchy before distortion; the cursor is on the first focus. In (b) the first focus region has been enlarged and the cursor is on the second focus region. (c) shows that the second focus region is also enlarged.

The First Law of Geography

However, the interface shown in Figure 5.13 is not a good focus + context interface since the clusters adjacent to the focus clusters have a lower level of detail than clusters far away from the focus. It violates the first law of geography.

Thirty years ago Waldo Tobler proposed what he called the *first law of geography* [Tob70]: Every thing is related to everything else, but closer things are more closely related. This is a law that can be applied to focus + context interfaces. For displays where there are geometrical relationships among the objects, the law can be understood as that

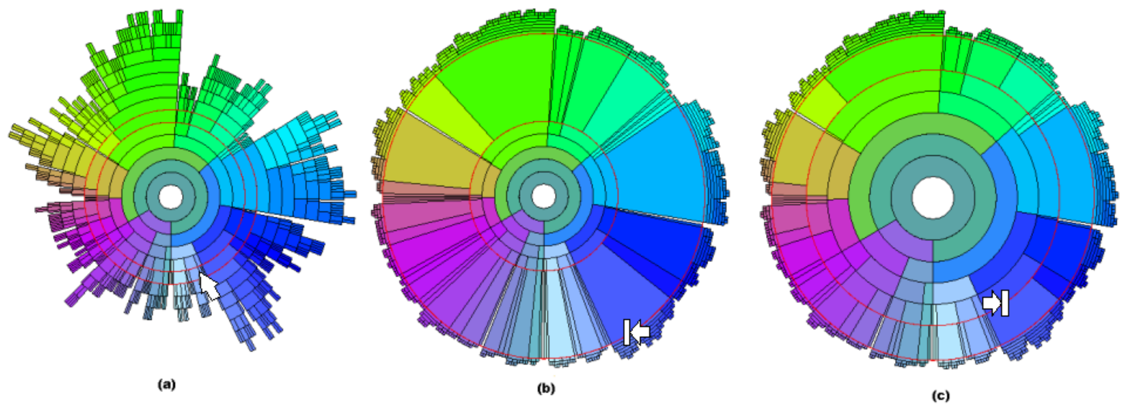


Figure 5.14: Radial Distortion Enlarging Inner Layers [YWR02]. (a) shows the hierarchy before distortion; the focus layer is outlined in red. In (b) the outer radial edge of the focus layer has been pushed outwards by decreasing the thickness of its outer layers. In (c) the inner radial edge of the focus layer has been pushed outwards to increase the thickness of its inner layers.

the objects closer to the focus objects in geometry have closer relationships to the focus objects than other objects. Thus they are more important context, and should be visualized in more, or at least no less, detail than the objects that are less close to the focus objects.

When there is no geometrical relationship among the objects, the law can still be applied with the geometry distances replaced by other dissimilarity measures. For example, when a document with many pages is browsed, the pages with page numbers closer to the focus page are usually considered more closely related to the focus page and visualized in more or no less detail than other pages. For another example, when objects are grouped into a hierarchy, the objects that are in the same cluster with the focus object are usually considered to be more closely related to the focus object and visualized in more or no less detail than objects in other clusters.

The Hierarchical Image Brower [HB98] is an example of hierarchy visualization to which the first law is applied, as shown in Figure 5.15. It presents a collection of images. The images are grouped into a hierarchy according to their style and place of origin. The focus image is displayed in the center of the display. It can be seen that images in the same cluster as the focus image are displayed using larger thumbnails than other context images.

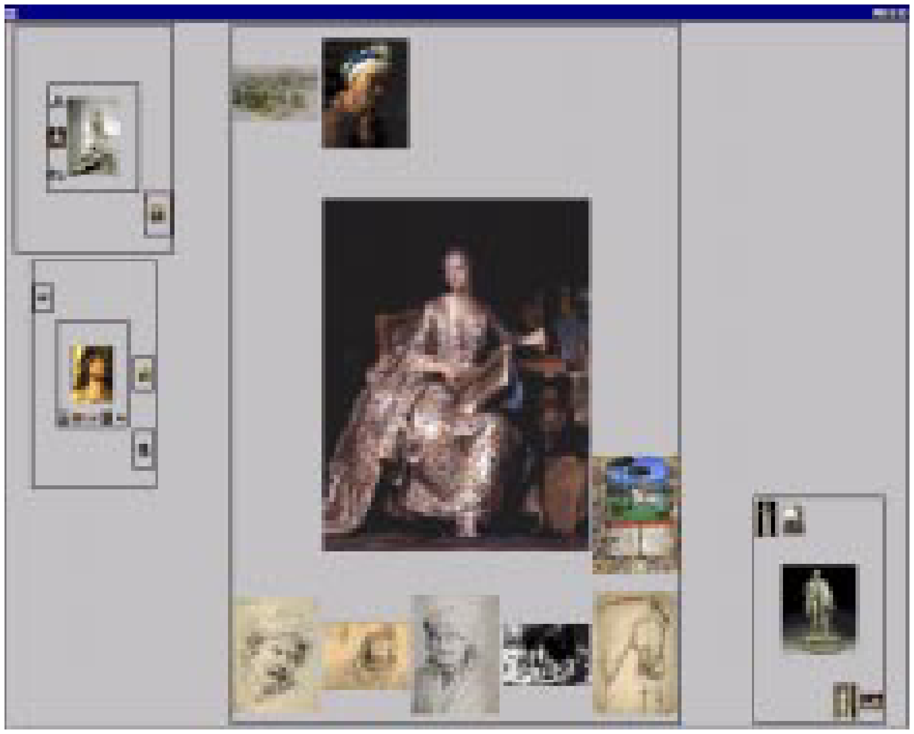


Figure 5.15: The Hierarchical Image Browser displays relevant images as thumbnails to provide context for a image displayed in more detail [HB98]. This figure is used without the authors' permission.

It needs to be noted that sometimes geometry distances give way to other dissimilarity measures that are considered more important than the geometry distances. For example, in a call map that visualizes many different callers in different areas, the person who is making a call to the focus person is often visualized in more detail than other persons who are closer to the focus person in geometry.

Focus and Context Placement

In many cases, there is no strict geometry relationship between the focus and context. For example, in a web page browser (such as the Zoom Brower [Hol97]), there is no geometry relationship among the focus page and the context pages. In these cases, the placement of the focus and the context is less restricted than the geometry constrained focus + context placement.

[Hol97] presents an interesting exploration of focus + context placement for linearly related objects in the context of web page visualization. In this exploration, the web pages are considered related only by the order of the pages, in other words, the page numbers. Thus the web pages are linearly related by the page numbers. [Hol97] discusses different approaches to placing the web pages, with one page visualized in a larger space than all the others. Figure 5.16 (a) shows the original page placement without a focus area. All pages are of the same size and ordered in a 2D space according to their page numbers that are marked within the pages.

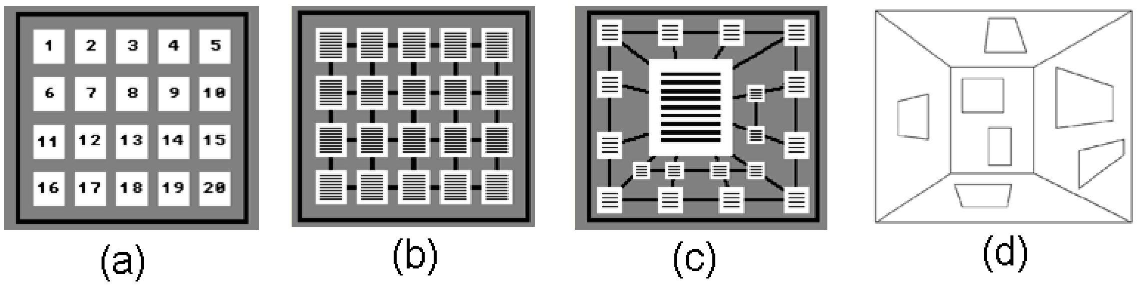


Figure 5.16: Placement of objects with 1D order [Hol97] inspired by the Document Lens. (a) Original page placement without a focus area. (b) Pages in (a) are linked by lines. (c) A focus page is magnified. (d) Illustration of the Document Lens. This figure is used without the authors' permission.

The first approach [Hol97] discussed was inspired by the Document Lens [RM93] that is illustrated in Figure 5.16 (d). The Document Lens is a 3D visualization for documents that allows users to focus on an area at a desired magnification. The presentation outside the focus is stretched to provide a continuous display of the global context. It is claimed in [Hol97] that the Document lens should not be applied directly because

- the spatial deformation which is applied to the text outside the focus may make it difficult to identify important features, and

- real-time performance may suffer because of the advanced calculations required for the display.

Thus the pages are linked by lines (Figure 5.16), and size reduction is applied to each page individually (Figure 5.16). However, it is found that in such a display, the pages closer to the focus are smaller than those further away, which is undesirable because the information closer to the focus is often regarded as more important for the context. Worse, the distribution of the non-focus pages is unclear - there is no easy correlation between the un-focused view and the focus view.

The second approach (Figure 5.17) was inspired by the rubber sheet view [SSTR93] but without the deformation of individual pages that would occur in such a display. This display has the advantage of a clear correlation between the page distribution in the non-focus and focus views, but it makes poor use of the display, with much space left empty.

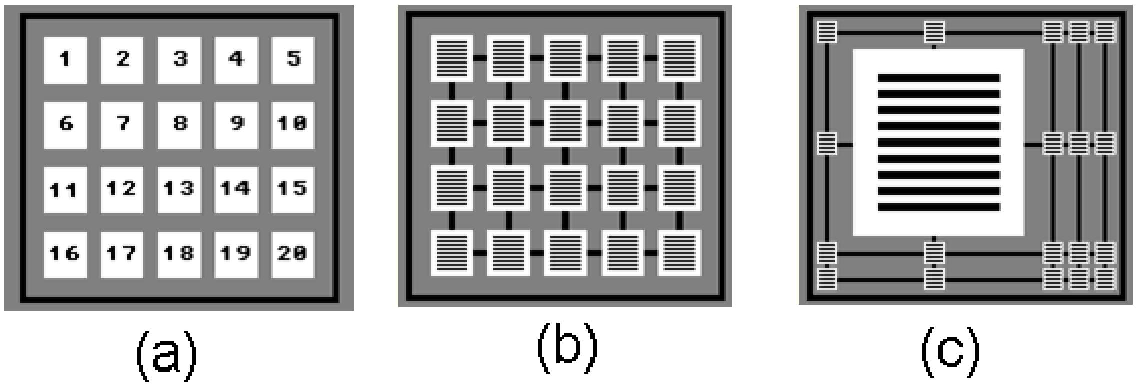


Figure 5.17: Placement of objects with 1D order [Hol97] inspired by the rubber sheet view. (a) Original page placement without a focus area (b) Pages in (a) are linked by lines. (c) A focus page is magnified. This figure is used without the authors' permission.

In the third approach, the focus page is placed approximately in the middle of the display, with the other pages arranged around it (Figure 5.18 (b)). The left-to-right, top-to-bottom ordering of the original display is kept, with all pages with lower page numbers placed above and to the left of the focus page, and pages with higher numbers placed below and to the right. This approach is called *flip zooming*. It has the following advantages:

- No deformation of the text outside the focus improves readability;
- Improved real-time performance since no advanced math is required for calculating the context display;
- Easier implementation of multiple foci.

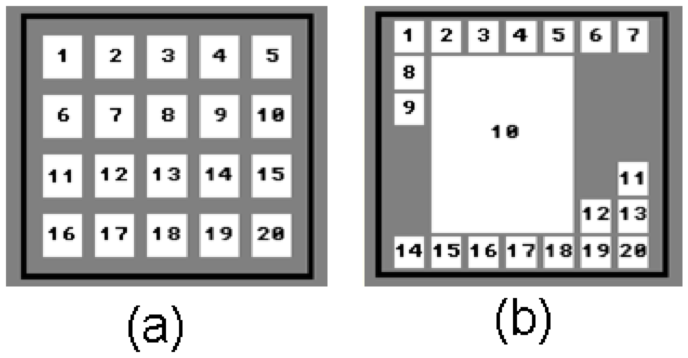


Figure 5.18: Flip zooming [Hol97]. (a) Original page placement without a focus area (b) A focus page is magnified. This figure is used without the authors' permission.

5.3.2 Distortion

There exist many distortion techniques, such as the fisheye lens [Fur86], the table lens [RC95], the perspective wall [MRC91], and the document lens [RM93]. We have studied some interesting applications of some of these techniques in traditional multi-dimensional display techniques [WY04]. For example, Figure 5.12 shows how we applied the idea of fisheye lens upon scatterplot matrices and got a focus + context scatterplot matrices display. Figure 5.19 shows how we applied the idea of perspective wall upon parallel coordinates and got a focus + context parallel coordinates display. While Figure 5.20 shows how we applied the table lens idea upon scatterplot matrices and got another focus + context scatterplot matrix display.

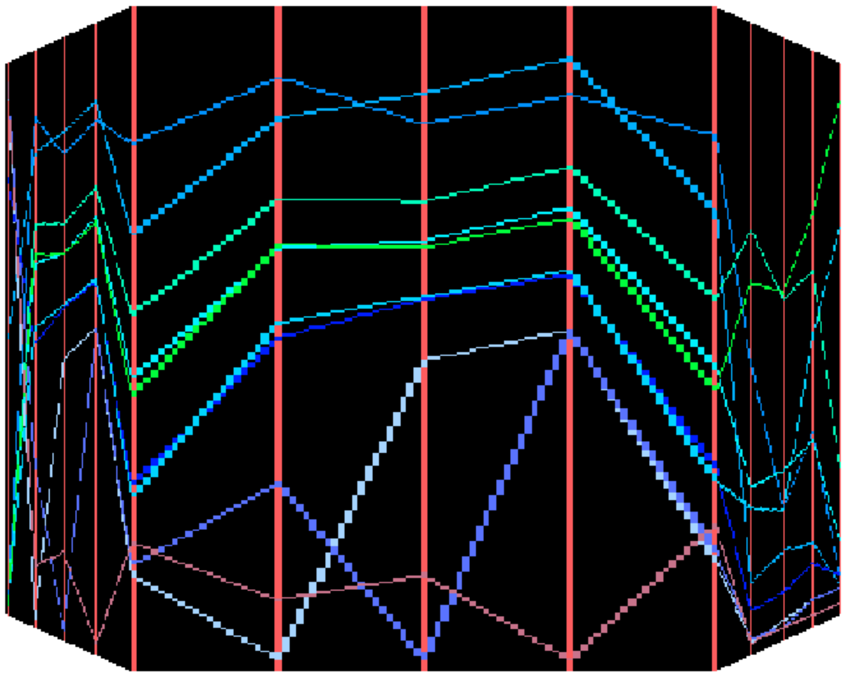


Figure 5.19: A focus + context parallel coordinates display [WY04].

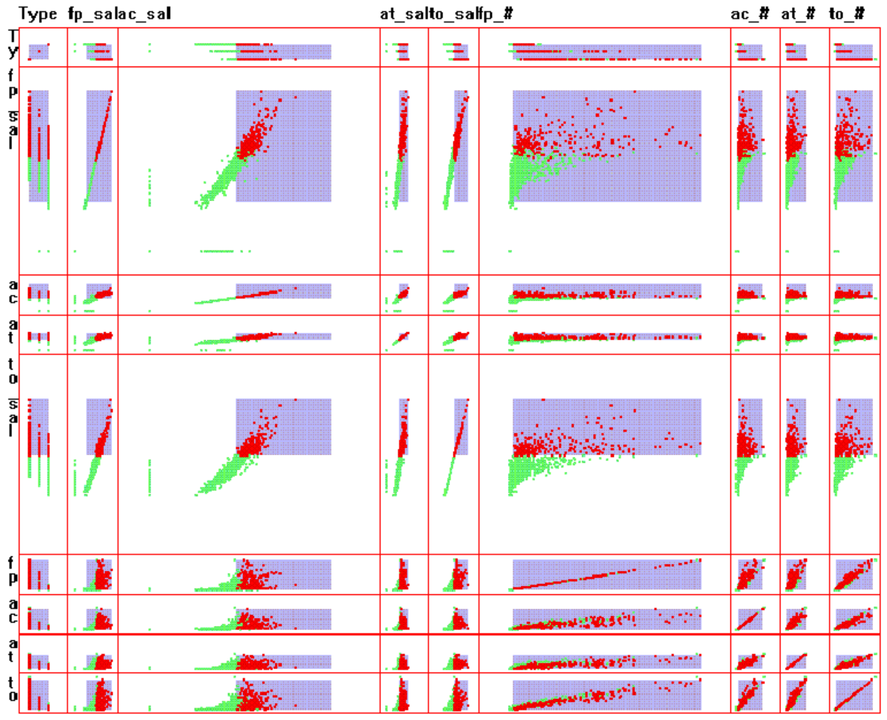


Figure 5.20: Another focus + context scatterplot matrix display [WY04].

5.4 Other Interactions

Besides zooming, panning, selection, and distortion discussed in the above sections, there are many other interactions common in multi-resolution visualization systems. I present some of them in the following sections.

5.4.1 Overlap Reduction

Most multi-resolution visualization systems handle large data sets. Despite the use of multi-resolution approaches, there still might be overlaps among objects in the display. In the following sections, I discuss some popular approaches to overlap reduction, and give a case study that uses these techniques in the VaR display [YPH+04].

Relocation

Definition: *relocation* - the operation that adjusts the location of objects in the display.

Overlaps can be reduced by relocation. For example, if we push two overlapping objects away from each other, the overlap between them can be reduced or eliminated. The advantage of relocation is that the objects need not be resized or reshaped, and no objects will be hidden or discarded from the display intentionally. The disadvantage of relocation is that this approach changes the topology of the display, which is often meaningful. For example, changing the positions of objects in a display generated by a proximity positioning operator, such as MDS or PCA, will reduce the accuracy of the visual depiction of the proximity information conveyed by the positions. The change of positions caused by relocation for this kind of display should be controlled carefully and possible inaccuracy of the visual depiction caused by relocation should be conveyed to the users.

Relocation can be performed manually or automatically. *Manual relocation* allows users to directly change the location of an objects or a group of objects in the display through mouse drag-and-drops, or other interactive methods such as entering the new position of an object through keyboard input. The advantage of this approach is that it is flexible and intuitive for users. A disadvantage of this approach is that the efficiency can be low if overlaps occur among lots of objects in the display. An example of manual relocation can be found in the case study presented in this section.

Automatic relocation automatically detects overlaps in a display, and reduces the overlaps by shifting positions of objects in the display while trying to keep the major topology of the positioning. Automatic relocation is also called *position shifting*. [War02] discusses several existing approaches that can be used for automatic relocation, such as Cleveland's random jitter approach [Cle93], and Keim and Hermann's quadtree approach [KH98]. Cleveland [Cle93] employed random jitter in statistical graphics where glyphs were positioned using their values in discrete dimensions, in which case overlaps could be serious without relocation. Keim and Hermann [KH98] used a quadtree structure to relocate points to maintain proximity to their original positions while eliminating overlaps.

Figure 5.21 shows an example of automatic shifting in a star glyphs display of the Iris data set. In Figure 5.21 (a), the glyphs are positioned according to their first two principal components. Some glyphs overlap among each other. In Figure5.21 (b), the positions of the glyphs are the result of an automatic shifting algorithm on the original positioning. Overlaps have been reduced in this display by pushing overlapping glyphs away from each other. The major topology is kept in this display, in other words, adjacent glyphs are still adjacent to each other and distant glyphs are still far away in general.

It needs to be noted that while a minor shifting probably will not significantly change the interpretation of the display, in cases where the object density is extremely high, an object's final position may be far from its initially assigned location [War02]. Thus the original topology of the positioning can be lost. So it is important to remind users of the

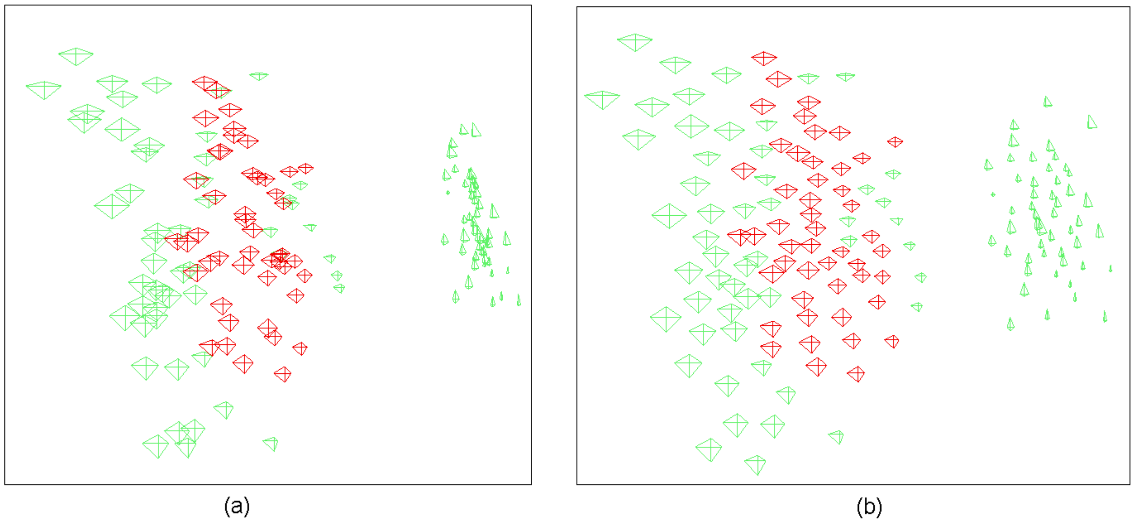


Figure 5.21: Star Glyphs Displays of the Iris Data Set. (a) Before automatic shifting, there are many overlaps among the glyphs. (b) After automatic shifting, overlaps among the glyphs have been reduced.

use of automatic shifting algorithms, and allow them to compare the positioning before and after the shifting.

Scaling

Definition: *scaling* - the operation that proportionally changes the sizes of all objects in a display.

Figure 5.22 illustrates how overlaps can be reduced through scaling. In Figure 5.22 (a), three rectangles overlap. In Figure 5.22 (b), the overlaps are diminished by shrinking the sizes of the rectangles proportionally. The centers of the rectangles are not changed. It can be seen that although the sizes of the rectangles change, the relative sizes of the rectangles remain the same.

Users can be allowed to interactively change the scaling factor (the extent to which the sizes are changed) to trade off between checking details of the objects and reducing overlaps.

For example, the Hierarchical Parallel Coordinates display [FWR99b, FWR99a] al-

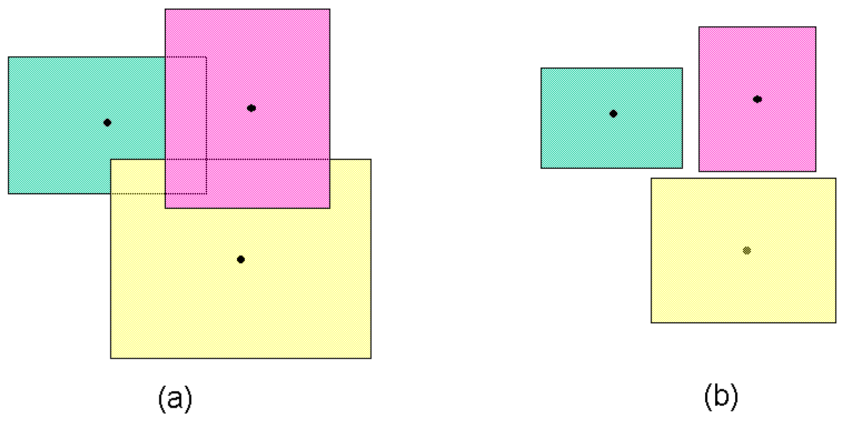


Figure 5.22: Scaling Illustration. (a) Three rectangles overlap. (b) The overlaps are diminished by scaling.

lows users to interactively change the extents of the bands through scaling to reduce clutter. Figure 5.23 (a) shows a hierarchical parallel coordinates display. All the bands are in their original sizes and the display is cluttered. Figure 5.23 (c) shows the same display after scaling the extents of the bands. The clutter is greatly reduced in this display.

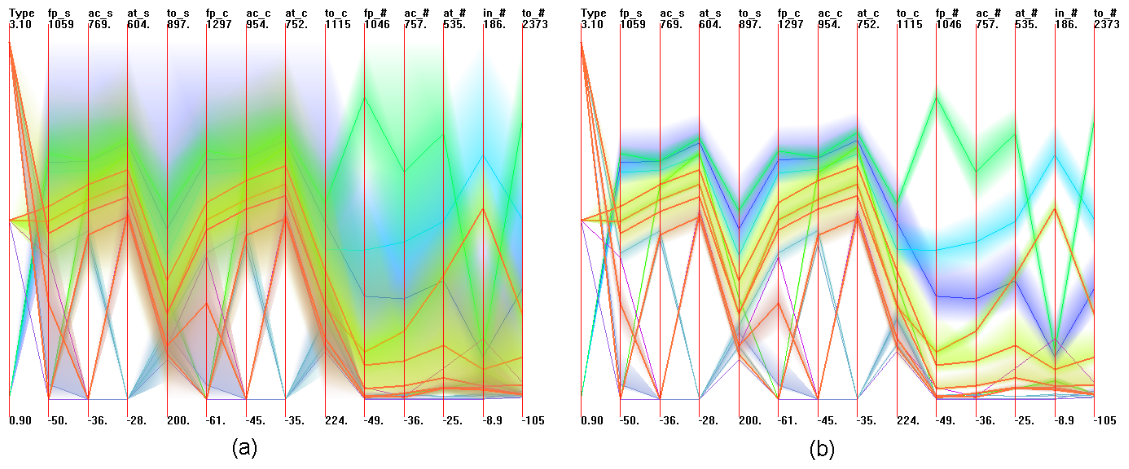


Figure 5.23: Extent Scaling in Hierarchical Parallel Coordinates. (a) All the bands are in their original sizes. (b) The extents of the bands are scaled to reduce clutter.

If some objects are scaled by a different factor than other objects, scaling turns into distortion.

Layer Reordering

Definition: *layer reordering* - the operation that changes the order in which objects are drawn in the display.

Layer ordering does not reduce overlaps, but it allows users to examine details that are hidden by the overlaps. Figure 5.24 illustrates how layer reordering allows users to view hidden objects. In Figure 5.24 (a), the green rectangle is drawn before the yellow one. Part of it is hidden by the yellow rectangle. In Figure 5.24 (b), the drawing order is reversed and the hidden part of the green rectangle is revealed in the display.

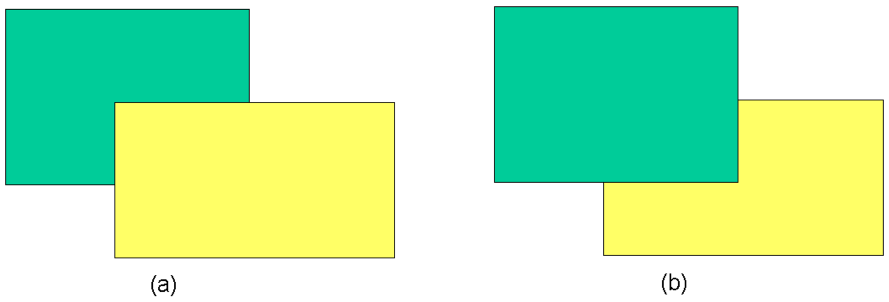


Figure 5.24: Layer reordering illustration. (a) The green rectangle is partially hidden by the yellow one. (b) The hidden part of the green rectangle is revealed.

Masking

Definition: *masking* - the operation that hides some objects in the display.

Masking is often used together with selection, i.e., either the selected objects or unselected objects are hidden from the display. An alternative to hiding the objects is to reduce the opacity of the objects. For example, the Hierarchical Parallel Coordinates display [FWR99b, FWR99a] allows users to interactively hide or change the opacity of brushed clusters or unbrushed clusters. Figure 5.25 (a) shows a Hierarchical Parallel Coordinates display. The brushed clusters are highlighted in red. Figure 5.25 (b) shows that the unbrushed clusters are hidden from the display, thus users can focus on the brushed clusters.

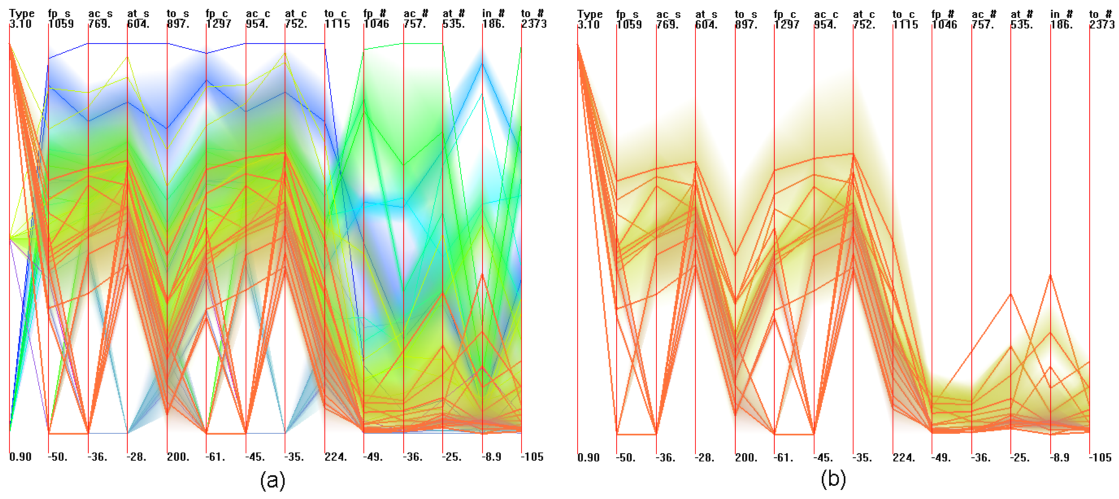


Figure 5.25: Masking in Hierarchical Parallel Coordinates. (a) Both brushed and unbrushed clusters are shown. Brushed clusters are highlighted in red. (b) Unbrushed clusters are masked.

Case Study: Overlap Detection and Reduction in VaR Displays

Value and Relation (VaR) displays [YPH$^+$04] are a multi-dimensional visualization technique for interactive exploration of data sets with hundreds of dimensions. The VaR displays provide a rich set of interaction tools. Many of them are overlap detection and reduction tools.

The VaR displays use pixel-oriented techniques to present values of data items in a high dimensional data set in a condensed manner. A glyph is generated for each dimension of the data set. These glyphs are positioned in a 2 dimensional space using MDS to approximate the proximity among the dimensions so that users can grasp the associations among dimensions. Glyphs of very similar dimensions can be very close to each other, thus overlaps happen in the VaR displays, as shown in Figure 5.26 (a). Overlaps can prevent a user from seeing details of an overlapped glyph. The VaR displays provide the following operations to overcome this problem:

Showing Names: By putting the cursor on the VaR display, the dimension names of all glyphs under the cursor position are shown in a message bar. Thus a user can be aware of the existence of glyphs hidden by other glyphs.

Layer Reordering: With a mouse click, a user can force a glyph to be displayed in front of the others. In this way, he/she can view details of a glyph originally overlapped. Figure 5.27 gives an example of layer reordering.

Manual Relocation: By holding the control key, a user can drag and drop a glyph to whatever position he/she likes. In this way a user can separate overlapping glyphs. Figure 5.28 gives an example of manual relocation.

Extent Scaling: Extent scaling allows a user to interactively decrease the sizes of all the glyphs proportionally to reduce overlaps, or to increase them to see larger glyphs. Figure 5.26 (b) gives an example of extent scaling.

Dynamic Masking: Dynamic masking allows users to hide the glyphs of unselected dimensions from the VaR display. In Figure 5.11, the glyphs of unselected dimensions are hidden using dynamic masking.

Automatic Shifting: This operation automatically reduces the overlaps among the glyphs by slightly shifting the positions of the glyphs. Figure 5.26 (c) gives an example of automatic shifting using a simple algorithm for reducing glyph overlaps borrowed from [War02].

5.4.2 Previewing

Definition: *preview* - a display generated of relative small cost that provides summarization information or simple judgments of a display to be generated.

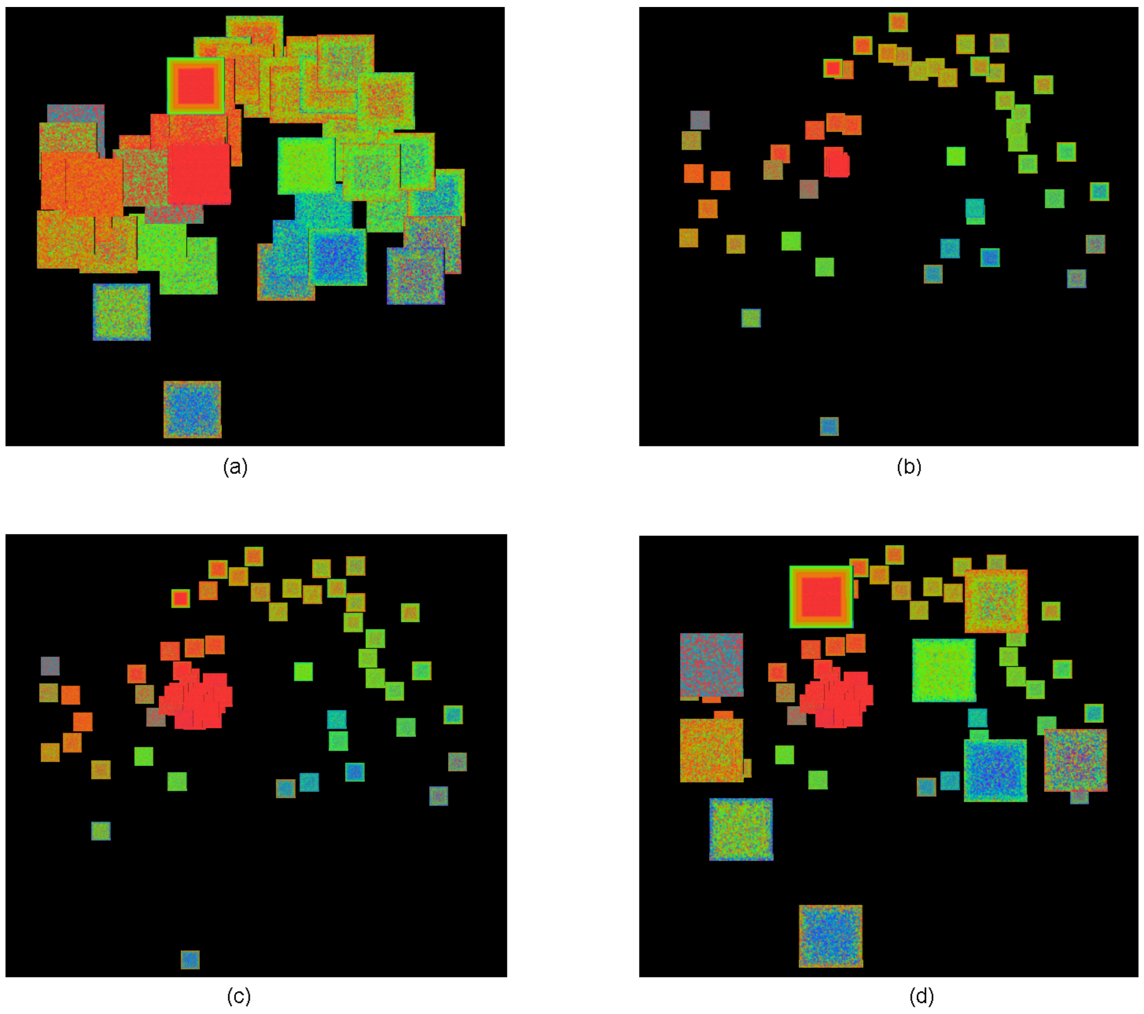


Figure 5.26: Extent Scaling, Automatic Shifting and Distortion in the VaR display [YPH$^+$04]. (a): A VaR display with seriously overlapped glyphs. (b) Overlap is reduced by decreasing the size of the glyphs. (c) Overlap of (b) is further reduced by automatic shifting. Notice that several glyphs appear in the center of the display which are previously non-visible in (b) due to overlaps. (d) Some glyphs are enlarged to examine detail within context. The data set is the Ticdata2000 data set (86 dimensions, 5,822 data items), which contains information on customers of an insurance company [vdPvS00].

When performing an exploratory task, users may encounter many useless displays before they reach a display of interest. For example, assume that we have a multi-resolution visualization system and a data set that can be divided into many subsets. The visualization system can provide the display of an overview of the whole data set and many displays of a detailed view, each of which contains a subset of the data set. Users can

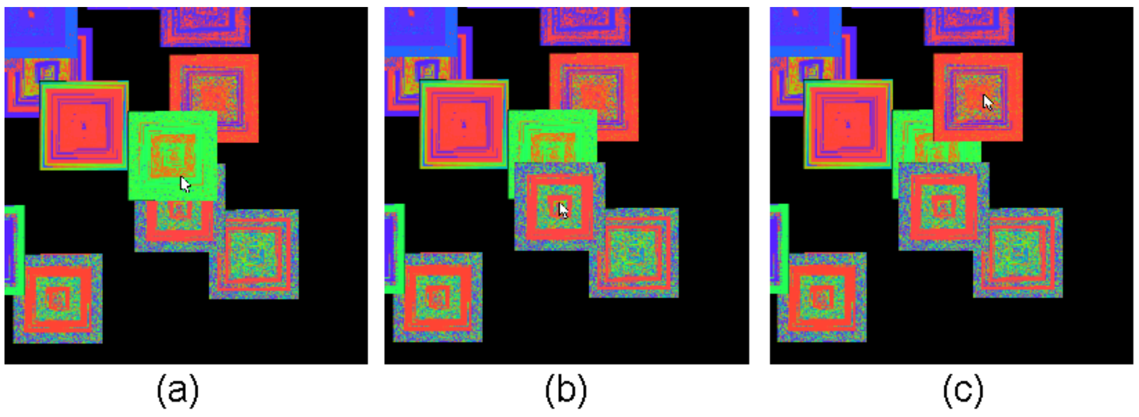


Figure 5.27: Layer reordering in the VaR display. (a) Three glyphs overlap in the center of the display. (a) The center glyph is drawn after its upper and lower glyphs. (b) The lower glyph is drawn after the center glyph. (c) The upper glyph is drawn after the center glyph.

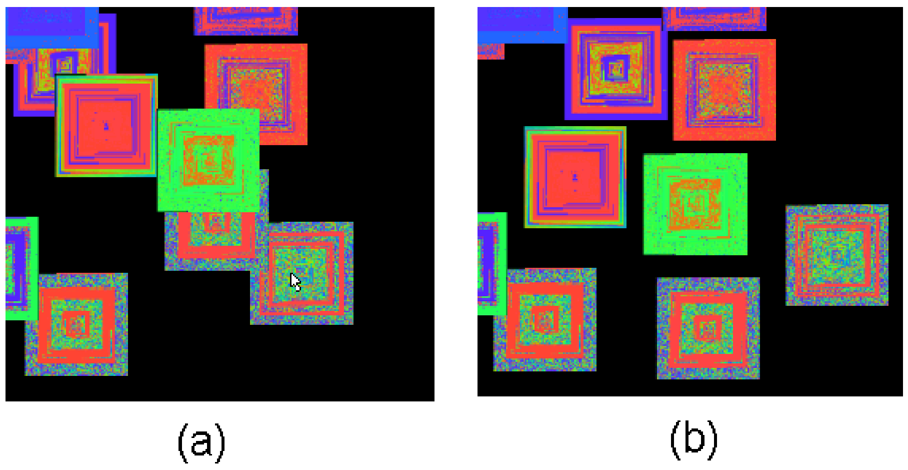
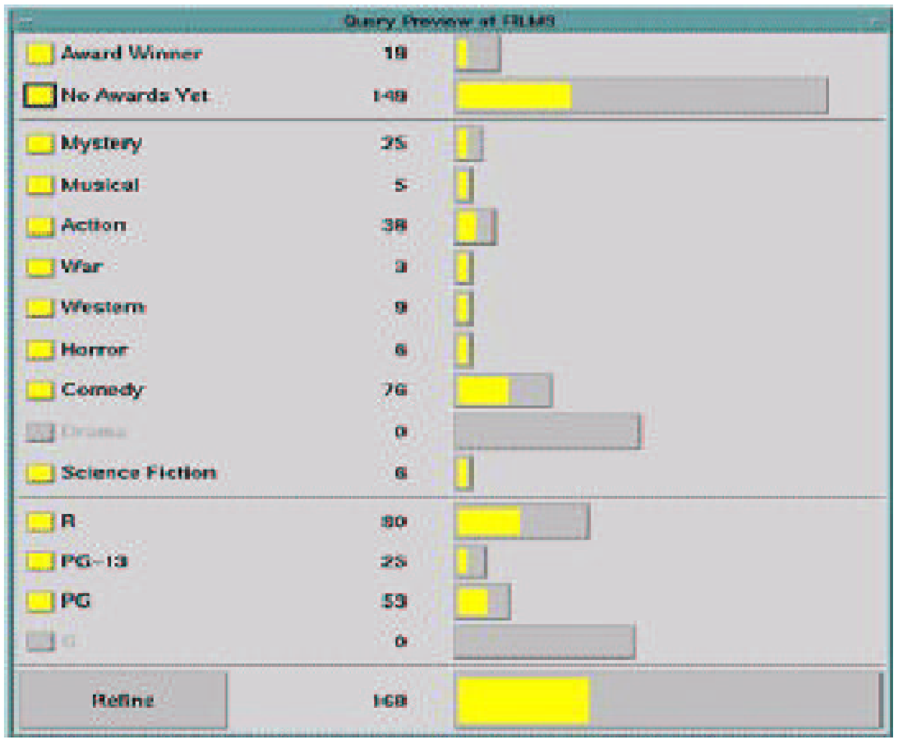


Figure 5.28: Manual relocating in the VaR display. (a) The glyphs overlap in the display. (a) Positions of some glyphs are manually changed. The overlaps are reduced.

jump into a detailed display by clicking its corresponding subset in the overview. If a user is looking for a detail that he/she does not know to which subset it belongs, he/she may load the overview, guess a possible subset, and jump to the detailed display from the overview. If his/her guess is wrong, he/she needs to go back to the overview and make a new guess. If the user can only know if his/her guess is correct after he/she sees the detailed display, the cost of a wrong guess can be expensive - it is often time-consuming to load a detailed display for a large data set. Previews try to minimize the cost of this process by providing users a kind of rough estimation before the detailed display is actually loaded.

Query previewing in dynamic queries [WS92, AS94, Shn94] is a typical example of using previews to improve efficiency. Dynamic queries provide a visual representation of query components and results. When they are used to query large distributed databases, slow network performance and limited local memory can became an obstacle. Users often waste time for queries that have zero-hit or mega-hit result sets. In order to solve this problem, query previewing was introduced into dynamic queries [DPS96]. Query previews supply data distribution information about the database that is being searched and give continuous feedback about the size of the result set for the query as it is being formed. The previews are visually presented to users. [TLH$^+$00] presents an empirical comparison that studied 12 subjects using dynamic queries with and without query previews. The result was that query previews sped up performance 1.6 to 2.1 times and led to higher subjective satisfaction. Figure 5.29 shows the query preview used in this user study.

Previews can be generated by applying the simplification operators (see Chapter 3) to the view (or part of a view) desired by the user. How to create a preview is closely related to the specific purpose of the previewing. For example, if the purpose of previewing is to prevent generating a cluttered display, a rough estimation of the number of objects in the display to be generated may be enough. It can be visualized in a small message bar or a small bar chart. For another example, if the purpose of previewing is to help users find a display with lots of clusters, the preview may need to be generated using the clustering operator (see Section 4.2). It can be visualized in a manner as complex as a tree visualization, or as simple as a message bar that reports the number of qualified clusters in the display to be generated.


Query Preview of FILMS

Award Winner	19
No Awards Yet	149
Mystery	25
Musical	5
Action	38
War	3
Western	9
Horror	6
Comedy	76
Drama	0
Science Fiction	6
R	80
PG-13	25
PG	53
G	0
Refine	168

Figure 5.29: The query preview used in a dynamic query system [DPS96]. The toggles on the left are used to define some rough ranges about the query that will be formed. The counts in the middle show the distribution of the result set (for the current settings of the toggles). The bars on the right also give the same count information (in a more visually recognizable form). The bar at the bottom shows the total number of hits. This figure is used without the authors' permission.

5.4.3 Animation

Animation is often used during zooming, panning and distortion operations. When animation is used, visual change in an interface is accompanied by a smooth transition that relates the old display to the new one. In other words, some other displays are inserted between the old display and the new one required by the user.

A commonly held belief is that animation helps users maintain object constancy and thus helps users to relate the two states of the system [BB99]. Robertson and his colleagues described this notion in their cone tree paper [RMC91] as "Interactive animation is used to shift some of the user's cognitive load to the human perceptual system. ... The perceptual phenomenon of object constancy enables the user to track substructure relationships without thinking about it. When the animation is completed, no time is needed for reassimilation".

This notion is supported by many user studies. For example, [BB99] reports a user study that examined how animating a viewpoint change in a spatial information system affected a user's ability to build a mental map of the information. The result of the user study showed that animation improved users' ability to reconstruct the information space, with no penalty on task performance. The reason for there being no penalty on task performance was that although animation cost extra in response time, users could relate the new representation to the old representation faster than without animation. Thus the time cost of animation was compensated.

Animation can be widely used in multi-resolution visualization systems. For example, when a user changes his/her viewpoint between views of different levels of detail, or between different sub-regions of the same view, animation can be used to reduce the user' cognitive load and help him/her relate the new representation to the previous one.

For example, [SZ00] uses animation to illustrate how the display is changed from an overview to a focus + context display in which the overview is shrunk and pushed to a corner of the display, and a detailed view of a sub-region of the overview is added to the display and occupies a large portion of the display. Figure 5.30 shows a sequence of snapshots of such an animation. Figure 5.30 (a) is the initial overview. In (b), the overview is shrunk and moved to the upper-right corner of the display. In (c), the focus area is highlighted and a small copy of it appears in a position close to the focus area in the overview. From (c) to (f), the copy is gradually enlarged and moved into the center of the display.

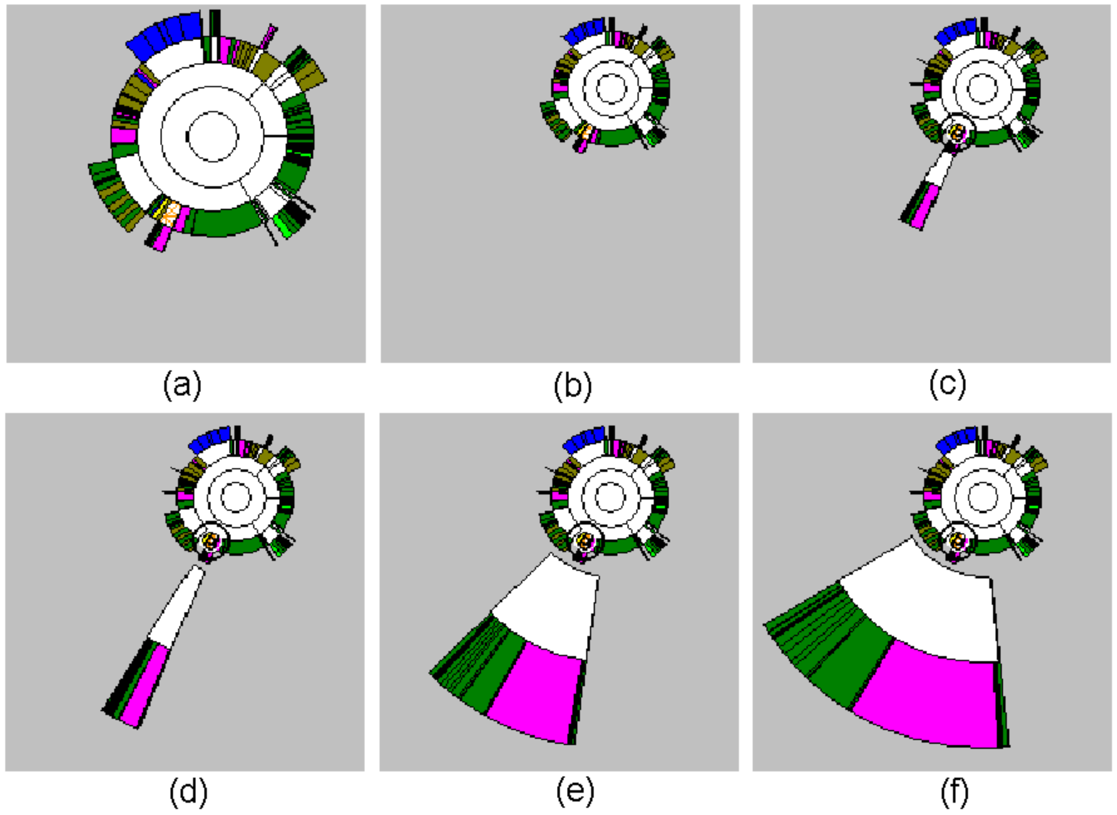


Figure 5.30: (a)-(f) is a sequence of snapshots that show how a display is changed from an overview to a focus + context display [SZ00]. (a) The overview (b) Overview is shrunk and moved to the upper-right corner of the display. (c) The focus area is circled and a small copy of it appears near the focus area. (d)-(f) The copy is gradually enlarged and moved into the center of the display. This figure is used without the authors' permission.

5.4.4 Dynamic Simplification

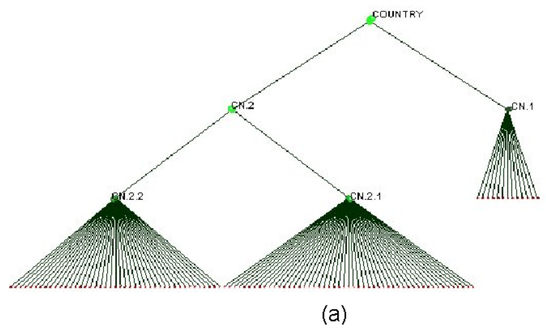
Definition: *dynamic simplification* - the process in which users interactively affect view simulation through interactions. It gives users more flexibility and allows them to use their domain knowledge to improve the simplification.

The Visual Hierarchical Dimension Reduction (VHDR) approach [YWRH03] is a typical example that uses dynamic view simplification. The VHDR approach is a multi-resolution visualization process that visualizes high dimensional data sets in lower dimensional spaces. The lower dimensional spaces are constructed by the users interactively. Dynamic simplification in VHDR is performed in the following manner. Firstly, VHDR clusters the dimensions and visually presents the constructed dimension hierarchy to users. Users can interactively navigate the dimension hierarchy to learn relationships of the dimensions. Then users can interactively select dimensions and dimension clusters of interest from the dimension hierarchy. After that, the data set is mapped into the lower dimensional space composed by the selected dimensions and dimension clusters and visualized. Since users can iteratively adjust their selection, multiple lower dimensional spaces can be generated by different user selections.

Pre-processing and auxiliary data structures can greatly accelerate dynamic simplification. I illustrate this point using the following examples.

The first example is the dynamic hierarchy construction approach [KLS00] that allows users to dynamically change the number of levels of hierarchies constructed for multi-resolution visualization. It has a pre-processing stage in which a binary dendrogram is built upon a data set using a hierarchical clustering algorithm. In the interactive-exploring stage, users can select different hierarchy levels. According to users' selections, hierarchies with different numbers of levels can be dynamically derived from the pre-computed binary dendrogram. Figure 5.31 shows two hierarchies for the same data set with different numbers of levels. The hierarchy in Figure 5.31 (a) contains three levels and presents an overview of the data set, while the hierarchy in Figure 5.31 (b) contains seven levels and presents a more detailed view of the same data set. The pre-computed dendrogram plays an important role in the approach. It enables the hierarchies to be constructed almost effortlessly without revisiting the original data set. Of course, pre-processing is needed and the pre-computed dendrogram needs to be stored. However, the response time in the dynamic hierarchy construction is much shorter than reclustering the data set without using a pre-computed dendrogram. This is a big gain in interactive exploration.

The second example is the dynamic sample selection approach [BCD03] that has been



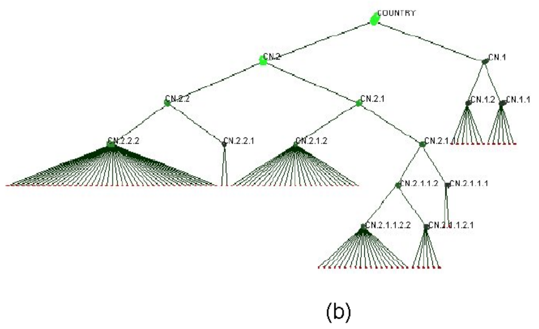


Figure 5.31: The dynamic Hierarchy example [KLS00]. (a) and (b) are two hierarchies derived from a same dendrogram. The hierarchy in (a) with 3 hierarchy levels provides an overview of the data set, while the hierarchy in (b) with 7 hierarchy levels provides more detail of the data set. This figure is used without the authors' permission.

discussed in Chapter 3. This approach provides appropriately biased samples for query estimation in the cost of pre-processing and storing pre-processed samples. The cost is trivial compared to the gain in dynamic processing efficiency.

Chapter 6

Usability Inspection for Multi-Resolution Visualization Systems

It has been stated that multi-resolution visualization systems are aimed at solving problems such as limited computational resources, restricted cognitive resources of human beings, and social issues such as protecting data security, shielding human privacy, and keeping peripheral awareness. Does a multi-resolution visualization really solve the problems for which it is intended?

There are many simplification operators and spaces upon which the operators can be applied. Which simplification is to be chosen if there are alternative simplification approaches? There are also multiple interfaces and interaction techniques that can be used by multi-resolution visualization systems. Which interface and interaction techniques are to be chosen? When there are multiple multi-resolution visualization systems available for the same application, which system should be chosen?

The cost of computational resources of multi-resolution visualization systems can be measured by computational experiments. However, for many other questions, the answer should be obtained from usability inspections.

In section 6.1, I briefly introduce two popular usability inspection methods, namely user testing (user study) and heuristic evaluation. In section 6.2, I present two user studies I conducted as examples of usability inspection.

6.1 Usability Inspection Approaches

Definition: *usability inspection* - the generic name for a set of methods based on having evaluators inspect or examine usability-related aspects of a user interface [NL94].

Visualization systems graphically present data and then allow the human to apply his or her perceptual abilities to make sense of the data. In particular, multi-resolution visualization systems are interaction-intensive visualization systems in which users interactively navigate among multiple views through interaction. Usability inspection is an important evaluation approach to assessing multi-resolution visualization systems.

There exists many usability inspection approaches, such as user testing, heuristic evaluation, feature inspection, heuristic estimation, consistency inspection, standards inspection, pluralistic walkthrough, and cognitive walkthrough [Nie95]. [Nie95] reports a survey of usability inspection methods. The participants of the survey were 42 persons who had taken a course on usability inspection methods 7-8 months before the survey. The participants were asked what methods they were in fact using and why they used or did not use the methods they had been taught. Figure 6.1 shows the results of the survey with regard to the frequencies of different methods used and the usefulness rating of the different methods given by the participants. Figure 6.1 reveals that user testing and heuristic evaluation are the most useful and most often used usability inspection methods. In the following sections, I discuss the user testing and heuristic evaluation approaches.

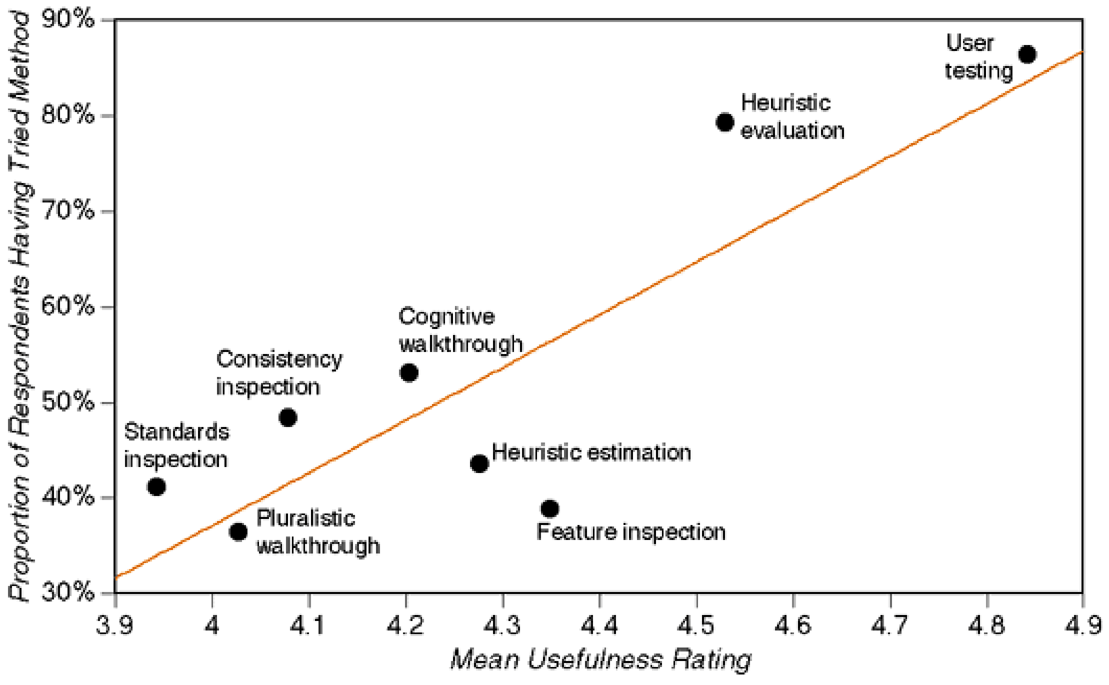


Figure 6.1: The result of a survey with regard to the frequencies of different methods used and the usefulness rating of the different methods given by the participants [Nie95]. This figure is used without the authors' permission.

6.1.1 User Testing

In user testing, participants (subjects) are real or representative users. They perform real tasks in a real work context.

User testing has several essential components: goal, subjects, information collection, information analysis, and experiment design. The goal is the motivation of a user test. For example, a user test can be aimed at comparing two visualization tools or evaluating how a visualization system performs for certain tasks. Hypotheses, such as "the new approach is better than the old approach", are often made before a user test. Subjects are users who perform real tasks using the visualization system evaluated. It is preferred that subjects are real users of the visualization system. Information is collected from a user test through experiment records and questionnaires. Usually before an experiment, the subjects will be asked to answer a questionnaire to report their background. After an experiment, the subjects will be asked to answer an “after” questionnaire which might includes multi-selection (or scaling) questions and open-ended questions to give feedback on the visualization systems they used. Subjects’ answers to the “after” questionnaires are an important measure to the system evaluated. Subjects’ often give valuable feedback in their answers to the open-ended questions if the questionnaires are well designed. The most important information is collected during the experiment, such as the time users spent to perform a task and patterns users detected and recorded, which I will discuss in detail in the following sections. In many user tests, video or other multi-media approaches are used to record subjects’ actions during the experiment so that they can be analyzed carefully. After information is collected from a user test, it needs to be analyzed in order to make decisions about whether or not the hypotheses are supported by the users study, and give suggestions on how to use or improve the visualization system evaluated.

I divide user tests of visualization system evaluation into two categories, namely *task-oriented user studies* and *open-ended think-aloud user studies* according to how the experiment is designed.

A typical task-oriented user study asks subjects to solved assigned tasks regarding the data visualized using the visualization system tested in a controlled manner. The assumptions of a task-oriented user study are:

- If the subjects solve a task quickly and correctly, the visualization system is suitable for the task.
- If the subjects need a lot of time or if the answers they provided are wrong, then the visualization system is not suitable for the task.

Subjects are often trained by several example tasks before the formal test. In the formal test, subjects are usually asked to solve the tasks as quickly as possible, and as correctly as possible. A *time-out* occurs if a subject decides that he or she cannot solve a given task and advances to the next task, and it should be handled differently from the other cases.

Quantitative analysis can be applied to the results of the user test, such as the correctness of the answers and the speed of task performance, to give an indication of how good the system is regarding the given tasks.

There are many examples of task-oriented user tests. For example, [GKS$^+$04] reported several user studies that evaluate the usability of a multi-resolution document repositories visualizer. Tasks such as locating a document or collection within the hierarchy, counting the number of documents contained within a collection, comparing the number of items contained within two separate collections, and counting the number of similar documents existing in the same collection to a given document were assigned to the subjects.

Different from the task-oriented user tests, open-ended think-aloud user tests ask subjects to search and report insights from a data set using the visualization tools after training. The definition and characteristics of an insight is defined as follows.

Definition: *insight* - an individual observation about the data by the participant, a unit of discovery [SND04].

Characteristics of an insight are [SND04]:

- Fact: The actual finding about the data. Distinct facts are counted for each participant.

- Time: The amount of time required to reach the insight. Initial training time is not included.

- Domain Value: The value, importance, or significance of the insight. For example, more global observations can be considered to be more valuable than trivial observations. The domain value can be coded on a certain scale for quantitative analysis by domain experts of the data.

- Hypotheses: Some insights lead users to identify a new hypothesis and direction of research. These are most critical because they suggest an in-depth data understanding.
- Breadth vs. Depth: Breadth insights present an overview of the data, but not much detail. Depth insights are more focused and detailed. This can be coded by domain experts.
- Directed vs. Unexpected: Directed insights are those that answer a specific question that the user was searching for. Unexpected insights are additional exploratory or serendipitous discoveries that were not specifically being searched for.
- Correctness: Some insights are incorrect observations that result from misinterpreting the visualization. This can be coded by domain experts and visualization experts together.

According to the characteristics of the insights, a set of measures can be designed, recorded, and analyzed in the user test in order to apply statistical analysis on the user test results and get useful indicators. For example, the average time used to find the first insight is a good indicator of how fast the subjects grasp the visualization tool. For another example, the total domain value of all insights found by subjects is a good indicator of if the visualization tool is useful for exploring the data visualized.

There are many examples of open-ended think-aloud user studies. For example, [SND04] reports an evaluation of microarray visualization tools. The subjects were asked to examine the data with the tool evaluated until they felt that they would not gain any additional insight. For another example, [YWR03b] reports a user test that compared two different multi-dimensional visualization approaches. The subjects were asked to find as many insights as they could from a data set using the visualization approach they were evaluating.

Formal user tests are often expensive since they often involve hours of concentrated work of many subjects. Once a user test fails because of improper process, training, test data, or tasks assigned, it is painful to run it again, in which case new subjects who are not involved in the previous one are often required in order to keep the “innocence” of the subjects to the system.

In order to prevent this from happening, pilot studies are often conducted before a formal user test in order to calibrate the evaluation methods, often with much fewer subjects involved. Pilot studies are often conducted in iterations after a user study plan is constructed, such as:

1. Conduct a pilot study according to the user study plan, with one or a few subjects.
2. If there are problems detected in the pilot study, improve the plan, go back to 1.
3. If there is no problem, proceed to the formal user test.

6.1.2 Heuristic Evaluation

Heuristic evaluation [NM90, Nie92, Nie94] is another form of usability inspection that is widely used to evaluate multi-resolution visualization systems. It is less expensive than a formal user test and can be conducted without a running prototype.

In a heuristic evaluation, a small set of evaluators examines the interface and judges its compliance with recognized usability principles (the “heuristics”). For example, in the context of multi-resolution visualization system, example heuristics can be that it should not take too many steps to go from the overview to the finest details, and that it should be easy for users to move from one sub-region to another without becoming lost. Nielsen has proposed 10 heuristics for Human-Computer Interaction [NM90]:

- visibility of system,

- match between system and the real world,
- user control and freedom,
- consistency and standards,
- error prevention,
- recognition rather than recall,
- flexibility and efficiency of use,
- aesthetic and minimalist design,
- help users recognize, diagnose and recover from errors, and
- help and documentation.

These heuristics are also essential for multi-resolution visualization systems. According to the specialty of multi-resolution systems, new heuristics can be created.

Heuristic evaluations are cheap, intuitive, and easy to motivate people to do them. They do not require much advanced planning and can be used early in the development process. To conduct a heuristic evaluation, general and category-specify heuristics, a running prototype or simple screen dumps, and three to five evaluators that have knowledge about usability problems are needed. During a heuristic evaluation, evaluators go through the interface and look for usability problems according to heuristics independently, with the experimenters available to answer domain specific questions. After a heuristic evaluation, the usability problems found should be rated for severity according to their frequency, impact and persistence.

Many case studies presented in the literature reporting the design and development of new visualization systems belong to heuristic evaluation. In these case studies, the authors usually propose some heuristics that the new systems are supposed to meet, then show examples of how the new systems do or do not meet these heuristics from the view of domain and visualization experts.

6.2 Examples

6.2.1 User Study 1: Assessing the Interactive Hierarchical Display Framework

The goal of our evaluation [YWR03b] was to assess if the Interactive Hierarchical Display (IHD) framework [YWR03b, YWR03a] is understandable by users, provides users with effective help in exploring large data sets as compared to traditional flat display techniques, and if interactive exploration tools such as the structure-based brush are useful and effective. Since several different multivariate visualization techniques can be embedded within the IHD framework, we selected one of them as a representative, namely, parallel coordinates, and then assessed it in both hierarchical and flat forms.

Experimental Methods and Setup

Our hypothesis was that, in general, subjects using the hierarchical parallel coordinates (HPC) could perform better in pattern finding for large data sets than subjects using the flat parallel coordinates (FPC). Patterns include clusters (isolatable subset of data that exhibit similar characteristics) and outliers (data items well separated from all the clusters). A steeper learning curve for HPC compared to FPC was expected since HPC is more complex than its flat counterpart. For one experiment, the test system was simplified to make available tools to the HPC and FPC subjects as comparable as possible. In HPC, only structure-based brushing and dynamic masking were provided to the subjects. In FPC, only data-driven brushing (painting over data to highlight it across all dimensions) and hiding and showing brushed versus unbrushed regions were available. We conducted two experiments in two different sessions, one with all subjects using HPC (HPC experiment) and the other using FPC (FPC experiment). The task of the subjects in both experiments was to find as many patterns as they could from the same data set using the provided tools.

Subjects

All the subjects in this evaluation were graduate students of the Worcester Polytechnic Institute. Figure 6.2 lists the number of subjects, their majors and backgrounds in visualization and data mining in the HPC and FPC experiment. Overall there were a larger proportion of computer science major students and a larger proportion of students with visualization or data mining background in the FPC experiment than in the HPC experiment. Besides those subjects, two computer science major graduate students with data mining background attended our pre-experiments, which were intended to test the proposed experiments for ease of understanding and level of difficulty.

Experiment	No. Subjects	CS-Majors	Prior Background
HPC	9	5	2
FPC	11	9	5

Figure 6.2: Information about subjects in evaluation experiments [YWR03b]. Prior background indicates how many had prior exposure to visualization or data mining.

Materials and Setup

To begin, an educational presentation was given to the subjects participating in each section (the presentation material can be downloaded from our website at `http://davis.wpi.edu/~xmdv`). Besides common aspects such as an introduction to multidimensional visualization, parallel coordinates and the characteristics of patterns such as clusters and outliers, the HPC presentation introduced HPC and structure-based brushing while the FPC presentation introduced FPC and data-driven brushing.

The HPC presentation was 20 minutes longer than the FPC presentation since it had more concepts to be introduced.

Both experiments used the Iris data set as the sample data set during the presentation and demonstration, and the AAUP (American Association of University Professors) Salary data set as the experimental data set to be explored by the subjects. The Iris data set is a relatively small data set, containing 4 attributes of 150 different breeds of the iris flower. The AAUP data set contains salary data for university professors, consisting of 14 dimensions and 1141 data items. Both are available from our website at `http://davis.wpi.edu/~xmdv`.

Both experiments were conducted in the same lab, where each subject used a 500 mhz Pentium III PC with 128 MB of RAM. The 'patterns' found by the subjects were saved into the local PC as pictures by the subjects during the experiments and collected together afterwards.

Procedure

Each experiment was divided into a training session and an experiment session. At the beginning of the training session, the presentation mentioned above was given to the subjects. Printed copies of the presentation were handed out to the subjects before the presentation as a handy reference to be used throughout the experiment. After giving the presentation, the coordinator of the experiment presented the visualization system using a computer with projection screen and demonstrated how to find patterns from the sample data set using the provided tools and how to save images of the patterns. Then the coordinator explained the meaning of the experimental data set. This session took about 30 minutes for the FPC experiment and about 50 minutes for the HPC experiment.

The experimental session was conducted immediately after the training session. The subjects were led to a PC with the visualization system already started. They were told to find and save as many ‘patterns’ as they could in 30 minutes (they were given the option of working longer if they wanted). After they finished this step, they were asked to answer an exit questionnaire and several open questions. Both experiments used the same questionnaire and open questions.

Before the formal experiments, we conducted two pre-experiments. In the first pre-experiment, a CS major graduate student with data mining background went through the FPC experiment followed by the HPC experiment. We noticed that he entered the pattern finding role soon in the FPC experiment, but kept wondering how to find patterns using the provided tools in the HPC experiment. This led us to conclude that we needed to provide a pattern finding strategy to the subjects of the HPC experiment. Then we designed such a strategy:

1. Keep the brushed region very small,
2. Set the brushed region to the maximum level of detail while keeping the unbrushed region at a high level of abstraction,
3. Move the brushed region around to search for patterns,
4. After a cluster is located, slowly increase the brushed region to find more data items belonging to the cluster until it is maximal,
5. If the pattern appears to be an outlier, slowly increase the brushed region to confirm that no similar data items are around it.

Using another CS major graduate student, we performed a second similar pre-experiment. The only difference from the first one was that the subject was taught the above strategy during the training section of the HPC experiment. This time the subject quickly entered the pattern finding role in the HPC experiment. Thus in the formal hierarchical parallel coordinates experiment, we taught this strategy to all the subjects. The results of the pre-experiments were not included in the experimental results.

Results and Discussion

As a first step, the ‘patterns’ saved by the subjects were analyzed by a visualization expert. The expert removed all the repeated ‘patterns’ by the same subject and bad ‘patterns’ that made no sense to him. Each pattern filtered out was numbered. The same pattern saved by different subjects received the same number. However, we only used these post-processed patterns in the parts of the analysis that were not time-related. The reason was that repeated patterns and bad patterns did not happen frequently and the time lines would be discontinuous if we considered them in time-related analysis.

Figure 6.3 shows the pattern finding results of the HPC and FPC experiment. The X axis of the figure lists all the patterns found in the experiments sorted by the average of the percentages of subjects that identified this pattern. The Y axis of the figure indicates the percentage of subjects who found the specific pattern in the HPC experiment or the FPC experiment. The solid purple curve and dashed blue curve represent the result of the HPC and FPC experiment respectively. The fact that the HPC curve lies above the FPC curve for most of the patterns means that higher percentages of subjects in the HPC experiment found those patterns than the subjects in the FPC experiment. This confirms our hypothesis that in general, subjects using HPC are more effective in identifying patterns in large data sets than subjects using FPC.

When we checked the patterns in detail, we found that HPC subjects were often looking for finer patterns due to the pattern finding strategy we recommended to them. The FPC subjects only focused on the large clusters and obvious outliers, since it is very difficult for them to find finer clusters and outliers hidden behind the overwhelming clutter of data. Actually, the FPC subjects only performed better in six of the 25 patterns, which are all large clusters or obvious outliers.

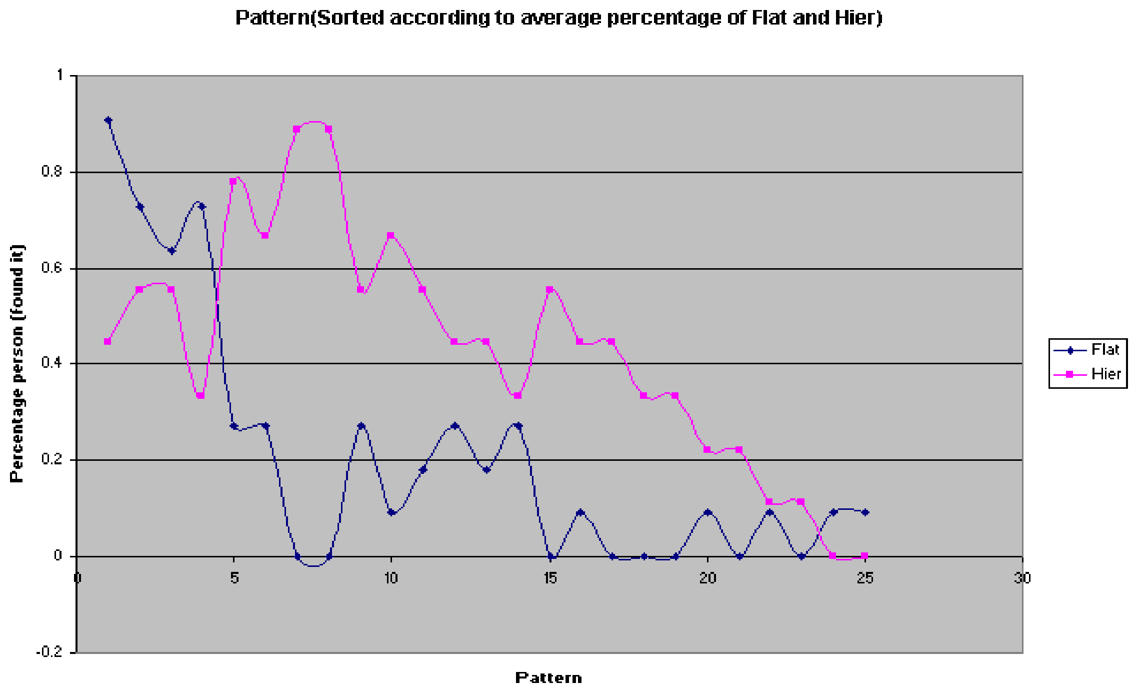


Figure 6.3: Pattern finding results of the HPC and FPC experiments [YWR03b].

Figure 6.4 shows the average time interval the HPC and FPC subjects used to find a pattern. A point(i, j) in this figure means that the subjects used an average time interval j to find their respective ith pattern. From the figure we can see that the HPC subjects used longer time to find their first two patterns. Having grasped the tools, they then found patterns at a higher speed than the FPC subjects. After the 9th pattern, their average speed of finding a pattern stayed below 3 seconds steadily. This indicates that HPC may help the subjects find patterns better than flat parallel coordinates once the subjects become familiar with the tools.

Figure 6.5 shows the individual subjects' process of pattern finding over time. The Y axis of the figures are the local times in the computers the subjects used. The X axis of the figures denotes the number of patterns saved by the subjects. In the first chart each line represents a subject in the FPC experiment, while in the second chart each line represents

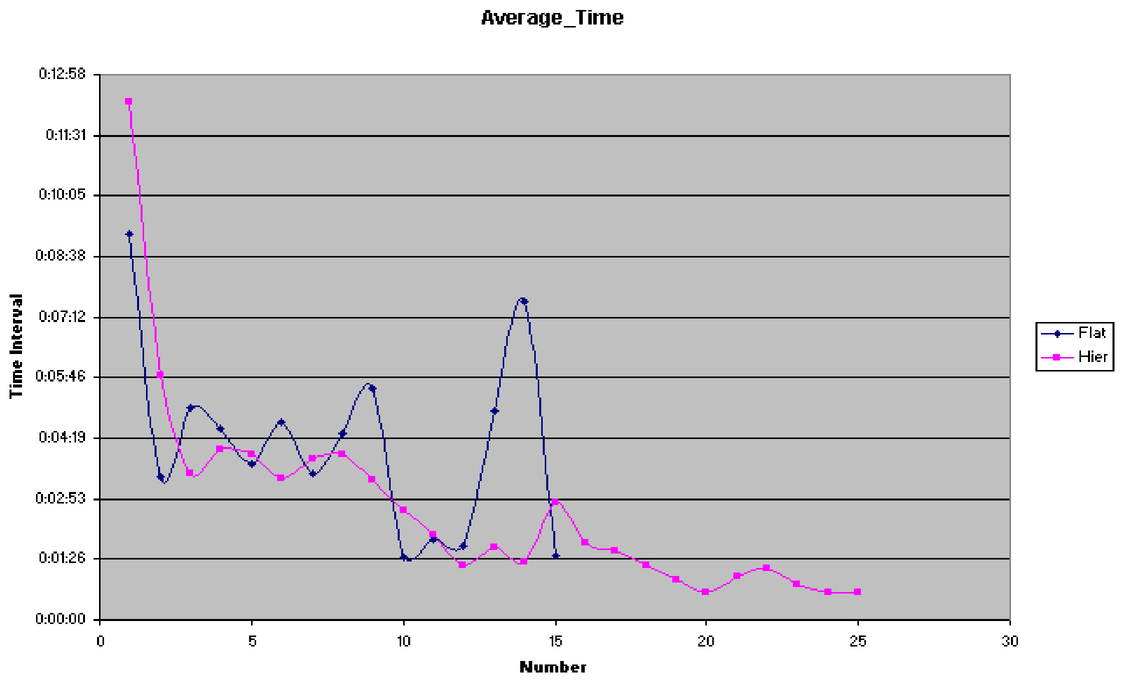
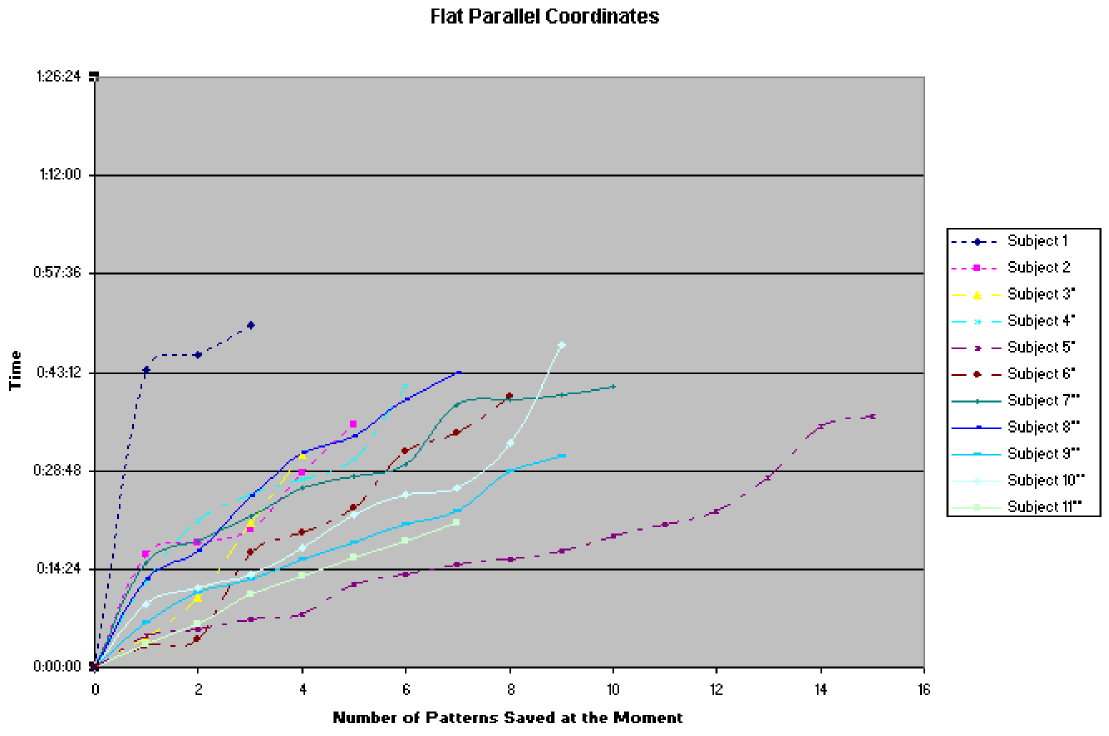


Figure 6.4: Average time interval in pattern finding in the HPC and FPC experiments [YWR03b].

a subject in the HPC experiment. It is obvious that there is a big variance of the subjects' performance in the FPC experiment; the subjects who were CS majors and had visualization or data mining background performed much better than other subjects. But in the HPC experiment, the variance is much smaller. Even more interesting, many non-CS major students with no relevant background performed better than some computer science major students with relevant background. A possible explanation for this phenomenon is that with the help of HPC, the difficulty of exploring a data set is reduced so that the background of the users becomes less important. Another possible reason could be that experienced users need to overcome existing strategy in order to use new ones.



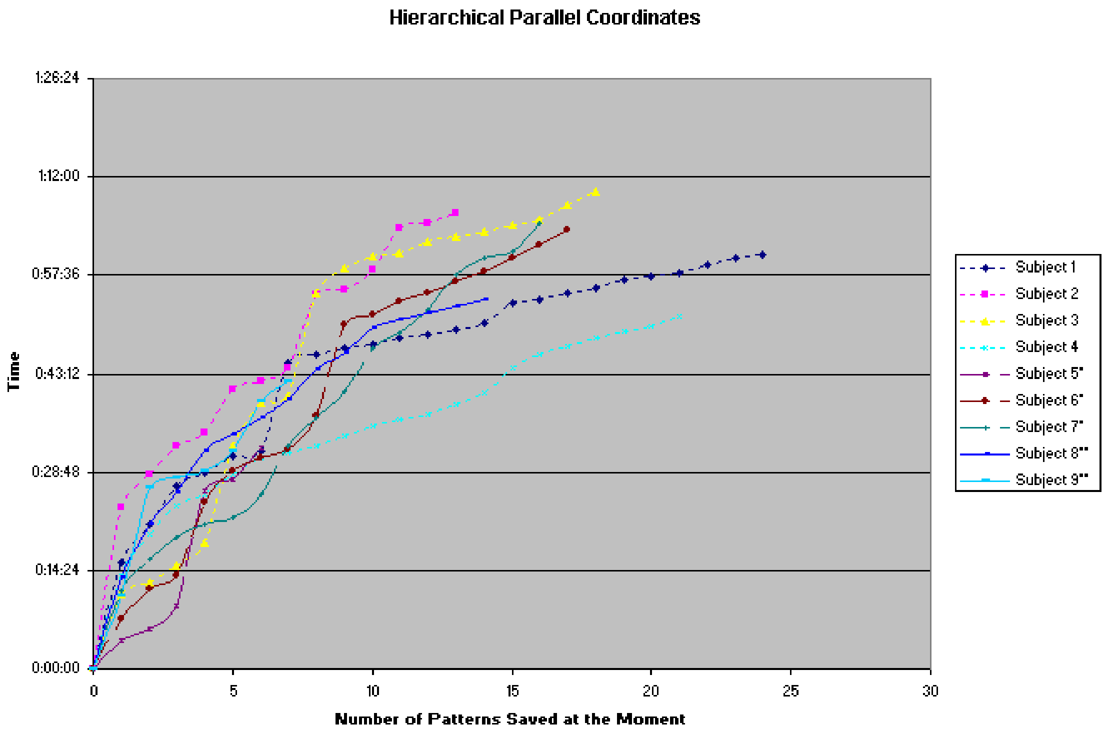


Figure 6.5: Subjects' process of pattern finding [YWR03b]. **: CS student with relevant background; *: CS student; No Mark: non-CS student without relevant background.

Open questions and Suggestions

The most interesting question among the open questions was “If you had more time, do you think that you could find more patterns?” Two of the eleven FPC subjects answered definitely no, and two others answered yes with hesitation. The numbers of patterns they found were 6, 9, 7 and 7 respectively. On the contrary, only one HPC subject answered no after he found 14 patterns. Many HPC subjects said that they searched the hierarchical tree in a certain direction and did not finish the searching yet when the time for the experiment had expired. They felt that they certainly could find more patterns if they spent more time.

The patterns found by the HPC subjects showed that the pattern finding strategy we recommended to them encouraged them to find finer clusters. But some obvious large clusters of the data set were often omitted using this strategy. We need to develop other HPC pattern finding strategies to help users grasp the overall trends of the data set, such as a top down strategy, which starts from brushing large clusters at high levels of abstraction, then gradually increases the level of detail and decreases the range of the brushed region.

The HPC subjects suggested that we make the structure-based brushing more flexible by allowing direct selection from the data display area. Also, they wanted to control the range of the brushed region through textual input to control it more precisely.

6.2.2 User Study 2: Comparing Alternative Distortion Approaches in InterRing

Background of InterRing

Hierarchy visualization has been widely studied in the information visualization area due to the ubiquitous existing of hierarchical data structures. Recent evaluations [SCGM00, BN01] of existing hierarchy visualization techniques reveal that the radial, space-filling (RSF) hierarchy visualization is a promising one among innumerous hierarchy visualization techniques. An RSF hierarchy visualization is generated using the following rules:

- Deeper nodes of the hierarchy are drawn further from the center;
- Child nodes are drawn within the arc subtended by their parents;
- The sweep angle of a leaf node is proportional to one of the node's properties or otherwise set to a uniform size;
- The sweep angle of a non-leaf node is the aggregation of all its children.

RSF techniques for hierarchy visualization have several advantages over traditional node-link diagrams, including the ability to efficiently use the display space while effectively conveying the hierarchy structure. However, it has the drawback that small slices in an RSF display for dense hierarchies are sometimes difficult to distinguish. Existing distortion techniques to address this issue [AH98, SZ00] have limitations, such as supporting only single focal points and reduced display space usage efficiency.

I designed a new distortion technique in our RSF system named InterRing [YWR02]. This technique allows multiple foci and does not reduce display space usage efficiency. Examples of this methods are shown in Chapter 5.

When designing this new distortion technique, I discovered that there were many groups of alternative distortion strategies I could adopt, such as applying distortion on sweep angles of the nodes (circular distortion) or radius of the layers (radial distortion), alternative selection strategies, alternative highlighting strategies, and alternative mouse usage strategies. For each group of alternative distortion strategies, should I choose one from the alternatives or provide all the alternatives to the users? I conducted evaluations to compare these alternatives in order to make the decision. In the following evaluation [YWRP03], the selection strategies in circular distortion were evaluated.

To help users to *snap* a node edge (select an edge of a node without needing to place the cursor accurately on it) to perform a distortion, I developed a *pinning* approach. After users click on a node, this node and all its ascendant nodes are highlighted and pinned in the sense that when performing a distortion on a pinned node, either one edge of it is fixed (one-direction distortion) or the center of the node is fixed (two-direction distortion). When the cursor is in a layer that contains a pinned node, it will automatically snap to one of the two circular edges of the pinned node. Thus selecting a circular edge of a pinned node is rather easy. The ascendant nodes of the clicked node are pinned together with the clicked node to provide multiple distortable nodes to users. An alternative to the pinning approach is to directly grab the node edge to perform a distortion operation, which I term a non-pinning approach. Since I already had a running system, I decided to have a formal user test to assess and compare the pinning and non-pinning approaches.

Experimental Methods and Setup

The hypothesis was that, in general, subjects using the pinning approach would be able to perform grabbing operations faster than subjects using the non-pinning approach, since the pinning function provided additional assistance to them. A steeper learning curve for the pinning approach than the non-pinning approach was expected since the pinning approach is more complex than the non-pinning approach.

The testing system was designed in a way that the subjects could concentrate on the grabbing operation. When a subject clicks a start button on the system interface, a hierarchy is displayed with one target sub-branch to be enlarged. The target sub-branch to be distorted by the subjects was colored in a distinguished way from other sub-branches. The task of the subject was to grab and drag the target sub-branch or its ascendant to enlarge the target sub-branch to twice of its original size. After finishing a task, the subject could click the start button to initiate the next task. The subject could also abandon the current task and enter the next task directly by clicking the start button if he/she is not able to successfully perform it using the available approach. The system automatically recorded the time between the subject clicking the start button and the target sub-branch reaching twice its original size as the time the subject used to perform that task. All other unrelated functions had been removed from the system to help the subjects concentrate on the main task.

There were 20 subjects in total. They were divided into two groups. In each group, there were two graduate students majoring in mechanical engineering (one from WPI, the other from UIUC), and the other eight were graduate students in computer science department of WPI. In each group, there were three females and seven males. All subjects had never used similar systems before the experiments. The first group used the non-pinning approach to perform a task set followed by another task set performed using the pinning approach. The second group used the pinning approach to perform a task set followed by another task set performed using the non-pinning approach.

All the experiments were conducted one by one (one subject, one experimenter). All the experiments were run on a PIII laptop. The window size of the display was 5.5inches by 5.5inches. Seventeen of the twenty experiments were conducted in the same quiet office. The other three were conducted in another quiet office.

Each experiment followed the following procedure:

1. The experimenter introduces the first distortion approach to the subject, and shows how to perform the task on two simple examples (about 6 minutes).

2. The subject performs a task set (12 tasks) (about 10 minutes).

3. The subject answers questionnaire 1, which is used to evaluate the first approach. (about 2 minutes)

4. The experimenter introduces the second distortion approach to the subject, and shows how to perform the task on two simple examples (about 2 minutes, since the second one is only slightly different from the first one).

5. The subject performs another task set (12 tasks, all different from tasks in the first task set, but each task has the similar difficulty as its counterpart in the first task set) one by one (about 10 minutes).

6. The subject answers questionnaire 1, which is used to evaluate the second approach. (about 2 minutes).

7. The subject answers questionnaire 2, which compares the two approaches they used. (about 2 minutes)

The average time cost per experiment was about 32 minutes.

Result Analysis

Figure 6.6 shows all the subjects' performance for each task, namely, the time the subjects used to finish each task. Figure 6.6 includes 4 plots. The two plots on the left side show the performance of the first group of subjects in their first section (using the pinning approach) and their second section (using the non-pinning approach). The two plots on the right side show the performance of the second group of subjects in their first section (using the non-pinning approach) and their second section (using the pinning approach). The X axes of the plots represent the task sequence ordered by the target sub-branch size and then by the tree size. The Y axes of the plots are time axes. Each subject is represented by a unique symbol and color as shown in the legends.

There were 8 cases in which the subject abandoned a task. For these cases, we assign them a maximum time, namely, 180 seconds in the figure. From the figure we can find out that there was one subject in the first group who abandoned three tasks in both the non-pinning approach and pinning approach section. In the second group, one subject abandoned one task in both the pinning approach and non-pinning approach section. Considering the total number of subjects (10 for each group), we view these two subjects

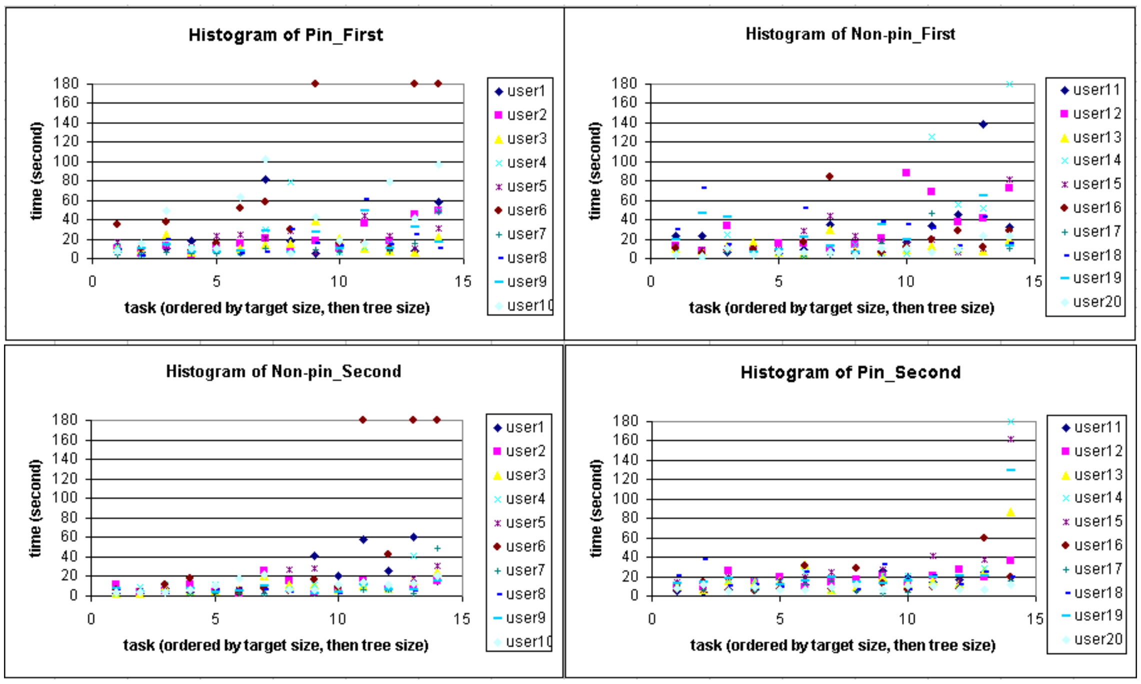


Figure 6.6: The subjects' performance for all tasks.

as exceptions. However, since they all performed in a normal way for the other tasks, we still take them into account for their finished tasks while omitting their performance in their abandoned task in our later analysis.

Figure 6.7 shows the average time of each group used to perform the tasks in the pinning and non-pinning sections. The X axis of the figure represents the task sequence ordered by the target sub-branch size and then by the tree size. The Y axis of the figure is time axis. There are four curves in this figure. Each has a unique color and symbol. Pin_first in the legend indicates the performance of the second group in its first section (using the pinning approach). Nonpin_first in the legend indicates the performance of the first group in its first section (using the non-pinning approach). Nonpin_second in the legend indicates the performance of the second group in its second section (using the non-pinning approach). Pin_second in the legend indicates the performance of the first group in its second section (using the pinning approach). From this figure we can determine the performance of the second group in the second section (using the non-pinning approach) wins in all the tasks.

The above fact overrides our hypothesis that the subjects using the pinning approach could perform grabbing operations faster than subjects using the non-pinning approach since the pinning function provides additional assistance to them. The reason is that using the pinning approach, the subject needs to perform two actions in order to grab an edge, namely, the pinning action by clicking the right mouse button and the grab and drag action by dragging and dropping the left mouse button. Using the non-pinning approach, only the second action is needed. It is true that the pinning action helps the subjects find the ascendants and possible distortion space. But the subjects of the second group had been trained on how to find the ascendants and possible distortion space with what they had learned from the first section (using the pinning approach). So in the second section (using non-pinning approach), they could do it fairly well without the help of the pinning action. Compared with the second group, the first group (using the non-pinning approach then the pinning approach) did not perform visibly different between their first and second sections. It also proves that the first pinning approach section provides a good training for the second group of subjects.

There is no big difference between the evaluations of the two approaches. The answers of the questionnaire 2 shows that both groups slightly appreciate the second approach they used. However, a popular opinion of the subjects is the pinning approach is informative and helpful for the novice users while unnecessary for the experienced users.

Conclusion

This evaluation suggests that for the experienced users, using the non-pinning approach as the grabbing strategy for the distortion is faster than the pinning approach. However, the pinning approach is a good training method for the novice users to help them grasp

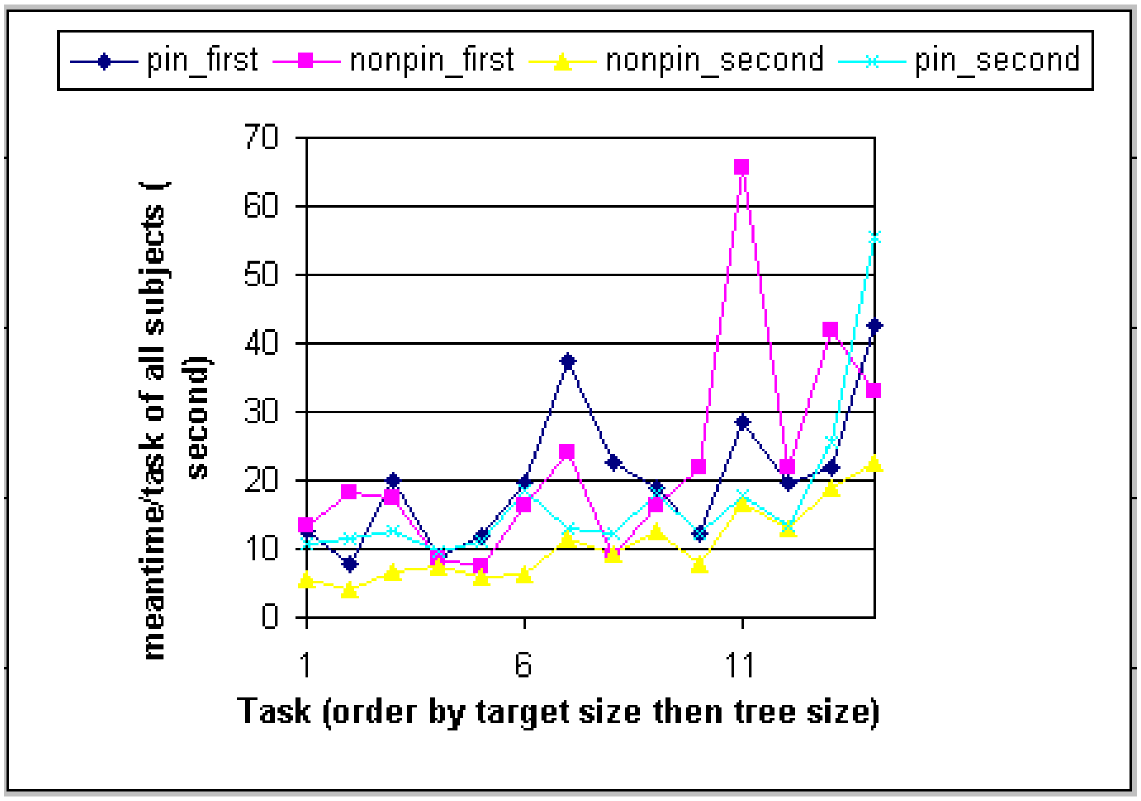


Figure 6.7: Average time each group used to perform the tasks in different sections.

the distortion operation.

Chapter 7

A Case Study

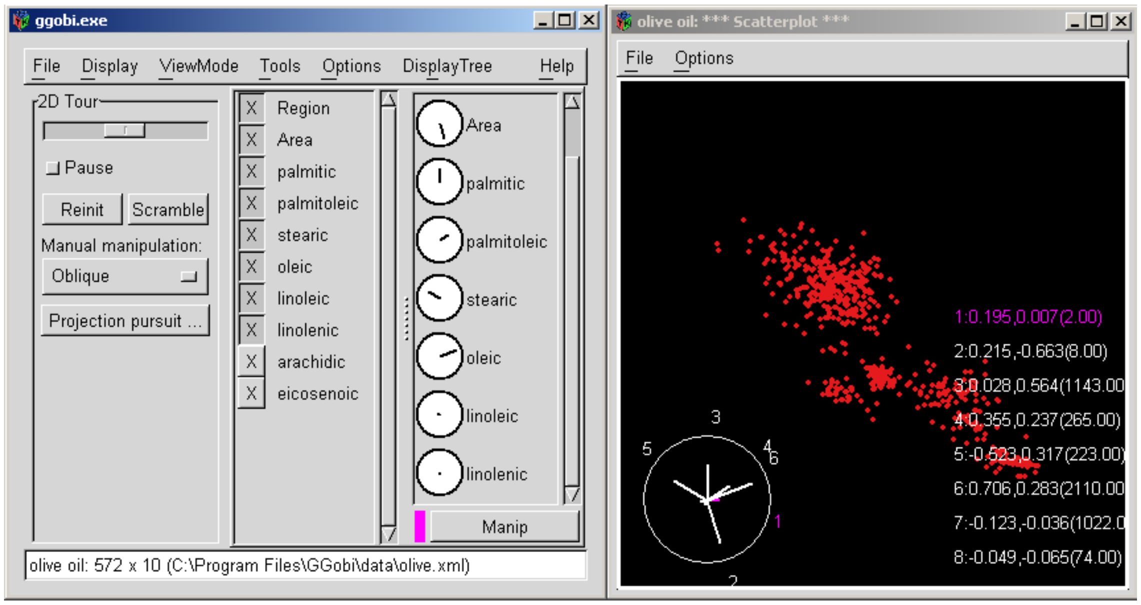
In this chapter, I present a case study that shows how to analyze and improve an existing multi-resolution visualization system using the proposed framework.

7.1 System Introduction

The system used in the case study is GGobi, an open source software distributed by AT&T. GGobi is a data visualization system for viewing high-dimensional data. It supports 2-D displays of projections of points and edges in high-dimensional space, as well as scatter-plot matrices, parallel coordinates, time series plots and bar charts. A set of interaction tools has been provided. Figure 7.1 shows the main control panel of Ggobi and scatterplot display. Figure 7.2 shows its parallel coordinates display.

7.2 Analysis

In my analysis, I will emphasize its multi-resolution visualization features and omit its other functions. GGobi used the following simplification approaches in data and visualization spaces:



```
olive oil: 572 x 10 (C:\Program Files\GGobi\data\olive.xml)
```

```
1:0.195,0.007(2.00)
2:0.215,-0.663(8.00)
3:0.028,0.564(1143.00
4:0.355,0.237(265.00)
5:-0.523,0.317(223.00)
6:0.706,0.283(2110.00
7:-0.123,-0.036(1022.0
8:-0.049,-0.065(74.00)
```

Figure 7.1: (a) The main control panel of GGobi. (b) The scatterplot display of Ggobi. The checked dimensions in the main control panel are mapped to the scatterplot display.

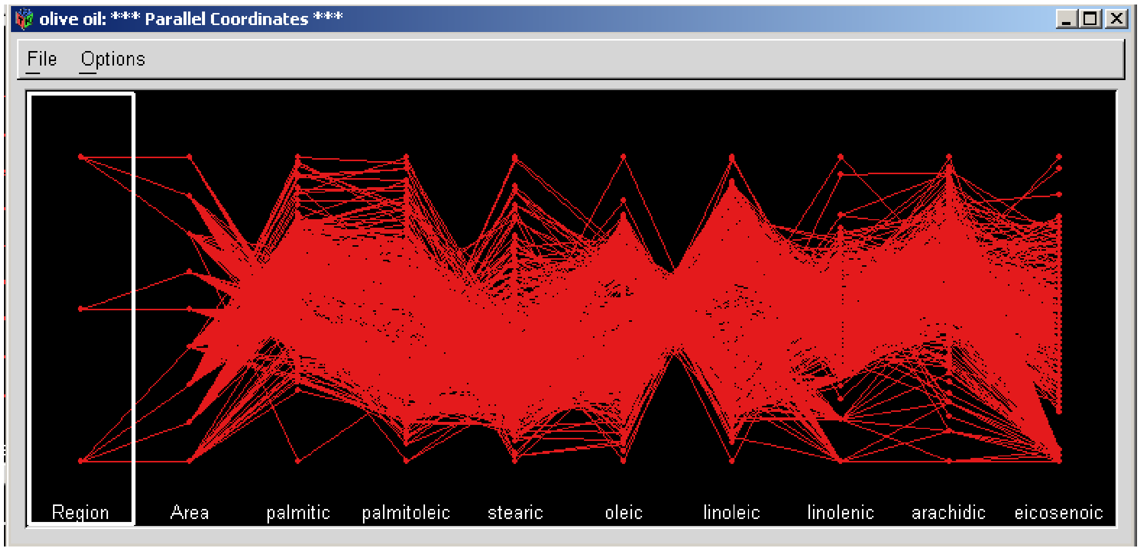


Figure 7.2: The parallel coordinates display of Ggobi.

- One-dimensional histogram in the data item space. Data distribution of single dimensions can be visualized in 1-D equi-width histograms.

- Sampling in the data item space. Users can visualize a sample of the data items rather than all the original data items using simple random sampling, uniform sampling, or other sample techniques provided by the system. Figure 7.3 shows the data item sampling interface.

- Generalization and sampling in the dimension space. Each dimension is generalized as a checkbox in the main control panel. Users sample the dimensions by clicking the checkboxes to visualize the data set in a lower dimensional space composed of the checked dimensions. Figure 7.1 shows that the checked dimensions are mapped to the scatterplot display.

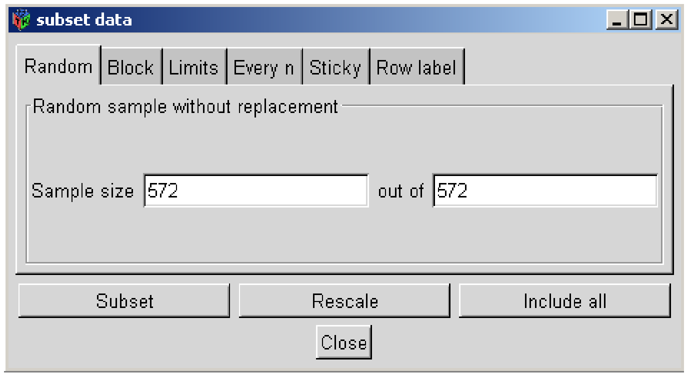
- Aggregation in the dimension space. There is a display type in which the selected original dimensions are projected into a 2D space. The projection in each dimension of the 2D space is a weighted sum of the original dimensions. Figure 7.1 shows the main control panel and the display in this mode. The selected dimensions and their weighting factors are visualized in the main control panel.

- Sampling in the visual encoding space. Users can filtering out some kinds of visual entities using interactions such as brushing.

- Generalization in the visualization structure space. Users can view display tree where each opened window is displayed as a node of the tree. They can then jump to any window by clicking its corresponding node.

Table 7.4 summaries the combinations of simplification operators and operands implemented and not implemented in Ggobi. A grid marked with * means that the combination of the operator and the operand is implemented in GGobi. The empty grids indicate the combinations that are not implemented in GGobi.

GGobi provides multiple windows to the users. For each individual window, zooming and panning are allowed, so that they are zoomable interfaces. Since multiple windows



Random sample without replacement

Sample size 572 out of 572

Figure 7.3: The sampling interface of Ggobi.

	item	dimension	topology	vis structure	visual encoding	screen
Sampling	*	*			*	
Clustering						
Histogram	*					
Other aggregation		*				
Wavelets						
Proximity Pos						
Generalization		*		*		
Others						

Figure 7.4: A grid marked with * means that the combination of the simplification operator and operand is implemented in GGobi.

are coordinated by data item IDs, they are also tied together to form an overview + detail interface.

Interactions such as selection, dynamic masking, automatic overlap-reduction are provided in GGobi.

7.3 Possible Improvements

Here I give a list of possible improvements for GGobi by comparing GGobi with the framework. Many of them are indicated by the empty grids in table 7.4.

- More simplification approaches, such as clustering and PCA, can be applied upon the data item space to help users discover patterns in large data set.

- More simplification approaches, such as clustering and MDS, can be applied upon the dimension space to help users create lower dimensional spaces easier and discover dimension relationships.

- Screen space simplifications can be introduced. For example, Fish eye lens can be provided to provide screen space simplification.

- Although users can use the multiple windows in GGobi as an overview + detail interface by setting different levels of detail for individual windows, it is troublesome to do so. It would be great if the system can automatically provide such an interface to the users.

Chapter 8

Conclusions

8.1 Summary

In this dissertation, I proposed a general framework for multi-resolution visualization systems. The essential feature of this framework is that a multi-resolution visualization system visually represents abstraction hierarchies of perceptions as views and allows users to interactively navigate within and among the views. The framework is composed of two major components:

View Simulation A multi-resolution visualization system simulates views through simplification of the data structure, or of the graphics generation processes if preferred perceptions are not provided to the visualization system.

Interactive visualization A multi-resolution visualization system visually presents the views to users and allows users to interactively navigate within and among the views.

I identified classes of view simulation approaches and described them in terms of simplification operators and operands (spaces). The simplification operators are further divided into four categories, namely the sampling operators, the aggregation operators, the approximation operators, and the generalization operators. I listed a large number of techniques falling into these categories that are used in existing multi-resolution visualization systems or are possible to be used in multi-resolution visualization systems. A large number of examples illustrate their applications in multi-resolution visualization systems.

The simplification operands (spaces) are also further divided into categories, namely the data space and the visualization space. How different simplification operators can be applied to these spaces is also illustrated using lots of examples.

Figure 8.1 summarizes the view simulation approaches.

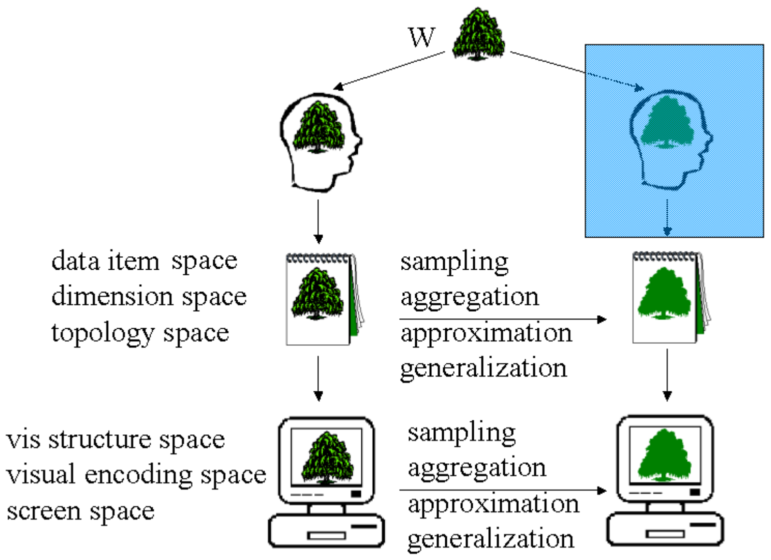


Figure 8.1: View simulation summarization. The blue box indicates that the boxed perception does not exist.

Table 8.2 lists the all possible combinations of operators and operands. The grids in the table marked by star indicate the combinations that are discussed in this dissertation. There are many empty grids in the table. Even for a starred grid, there are likely techniques belonging to that operator whose application on that space are not discussed in this dissertation. For the combinations not discussed, there are two cases. In the first case, there exist visualization systems that use the combinations, and they are not discussed in this dissertation. For example, sampling has been applied to the screen space in many applications. In the second case, the combinations are never used by any existing visualization system. Exploration of these may lead to new visualization techniques. For example, to the best of my knowledge, there is no existing visualization system applying histogram operators to the dimension space. When there are thousands or more dimensions, such techniques might be useful.

	item	dimension	topology	vis structure	visual encoding	screen
Sampling	*	*				
Clustering	*	*	*			
Histogram	*					
Wavelets	*		*			
Proximity Pos	*	*				
Generalization	*			*		
Others	*	*			*	*

Figure 8.2: Combinations of Simplification Operators and Operands. A grid marked with * means that the combination is discussed in the dissertation.

I presented three types of multi-resolution visualization interfaces, namely the zoomable interface, the overview + context interface, and the focus + detail interface. I also discussed common interaction tools used in multi-resolution visualization systems, such as zooming and panning, selection, distortion, overlap reduction, previewing, animation, and dynamic simplification. Examples were also widely used in those discussions.

The case study I described in Chapter 7 gives an example of how to analyze and improve existing multi-resolution visualization systems using this framework.

8.2 Contributions

The major contribution of this dissertation is not to cover all existing multi-resolution visualization techniques (though I tried to cover as many in the Information Visualization field as I could), or to present guidelines researchers must follow when they analyze, evaluate, or design a multi-resolution visualization system. The major contributions of this dissertation are:

- The framework proposed in this dissertation summarizes common features shared by a wide variety of multi-resolution visualization systems. It can help readers analyze and evaluate existing multi-resolution visualization systems.

- A wide variety of simplification techniques from different fields are presented in this dissertation, and their usage in multi-resolution visualization systems are discussed.

- A wide variety of multi-resolution visualization interfaces and interactions are presented and illustrated in this dissertation, and their usage in multi-resolution visualization systems are discussed.

- This dissertation can help readers design new multi-resolution visualization systems. There are many operators, operands, interfaces, and interactions presented in this dissertation. Some combinations of them have never been seen in the existing multi-resolution visualization systems. Try them and something may happen, who knows!

8.3 Future Work

In the future, I'd like to:

- Continue to evaluate and improve this framework by analyzing, evaluating, and designing multi-resolution visualization systems using it.

- Extend this framework to cover more visualization fields. As I have mentioned, most of the visualization systems I analyzed for generating this framework and examples used for illustrating this framework are in the scope of information visualization. However, I believe that such a framework will also benefit other visualization fields besides information visualization. In order to make this framework a better reference for other fields, more examples and techniques from those fields need to be analyzed and included into this framework.

- Implement many ideas inspired by this dissertation. In the dissertation, there are many places where I pointed out potential usage of some techniques in multi-resolution visualization systems, such as using multi-dimensional histograms in multi-dimensional visualization. It would be interesting to implement these ideas and evaluate them in real applications.

Bibliography

[ABKS99] M. Ankerst, M. Breunig, H.-P. Kriegel, and J. Sander. Optics: Ordering points to identify clustering structure. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 49–60, 1999.

[AC99] A. Aboulnaga and S. Chaudhuri. Self-tuning histograms: Building histograms without looking at data. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 181–192, 1999.

[Agg01] C. Aggarwal. On the effects of dimensionality reduction on high dimensional similarity search. *PODS*, 2001.

[AGGR98] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 94–105, 1998.

[AGP00] S. Acharya, P. Gibbons, and V. Poosala. Congressional samples for approximate answering of group-by queries. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 487–498, 2000.

[AH98] K. Andrews and H. Heidegger. Information slices: Visualising and exploring large hierarchies using cascading, semicircular discs. *Proc. IEEE Symposium on Information Visualization, Late Breaking Hot Topics*, pages 9–12, 1998.

[And72] D. Andrews. Plots of high dimensional data. *Biometrics*, 28:125–136, 1972.

[AS94] C. Ahlberg and B. Shneiderman. Visual information seeking using the filmfinder. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, 2:433, 1994. Color plates on page 484.

[Bas99] W. Basalaj. Incremental multidimensional scaling method for database visualization. *Proc. of Visual Data Exploration and Analysis VI, SPIE*, 3643:149–158, 1999.

[Bas01] W. Basalaj. Proximity visualisation of abstract data. Technical Report UCAM-CL-TR-509, University of Cambridge, Computer Laboratory, 15 JJ Thomson Avenue, Cambridge CB3 0FD, United Kingdom, phone +44 1223 763500, January 2001.

[BB99] B. Bederson and A. Boltman. Does animation help users build mental maps of spatial information? *Proc. IEEE Symposium on Information Visualization*, pages 28–35, 1999.

[BCD03] B. Babcock, S. Chaudhuri, and G. Das. Dynamic sample selection for approximate query processing. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 539–550, 2003.

[BCG03] N. Bruno, S. Chaudhuri, and L. Gravano. Stholes: A multidimensional workload-aware histogram. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 211–222, 2003.

[BCLC97] D. Brodbeck, M. Chalmers, A. Lunzer, and P. Cotture. Domesticating bead: Adapting an information visualization system to a financial institution. *Proc. IEEE Symposium on Information Visualization*, pages 73–80, 1997.

[Ber02] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[BETT99] G. Battista, P. Eades, R. Tamassia, and I. Tollis. *Graph Drawing: Algorithms for the Visualization of Graphs*. Upper Saddle River, N.J: Prentice Hall, 1999.

[BHR99] S. Björk, L. Holmquist, and J. Redström. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. *Proc. IEEE Symposium on Information Visualization*, pages 53–60, 1999.

[BKW89] A. Bruggemann-Klein and D. Wood. Drawing trees nicely with tex. *Electronic Publishing*, 2(2):101–115, 1989.

[BN01] T. Barlow and P. Neville. A comparison of 2-d visualization of hierarchies. *Proc. IEEE Symposium on Information Visualization*, pages 131–138, 2001.

[BS02] Z. Bićanić and R. Solarić. Cartographic generalization and contributions to its automation. *Geoadria*, 7(2):5–21, 2002.

[BW90] D. Beard and J. Walker. Navigational techniques to improve the display of large two-dimensional spaces. *Behav. Inform. Techn.*, 9(6):451466, 1990.

[CC94] T. Cox and M. Cox. *Mutidimensional scaling*. Chapman & Hall, 1994.

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

[Che03] H. Chen. Compound brushing. *Proc. IEEE Symposium on Information Visualization*, pages 181–188, 2003.

[Chu98] M. Chuah. Dynamic aggregation with circular visual designs. *Proc. IEEE Symposium on Information Visualization*, pages 35–43, 1998.

[CK95] J. Carriere and R. Kazman. Interacting with huge hierarchies: Beyond cone trees. *Proc. of Information Visualization '95, p. 74-81*, 1995.

[CK03] P. Craig and J. Kennedy. Coordinated graph and scatter-plot views for the visual exploration of microarray time-series data. *Proc. IEEE Symposium on Information Visualization*, pages 173–180, 2003.

[Cle93] W. Cleveland. *Visualizing Data*. Hobart Press, 1993.

[CM88] W. Cleveland and M. McGill. *Dynamic Graphics for Statistics*. Wadsworth, Inc., 1988.

[CMS99] S. Card, J. Mackinlay, and B. Shneiderman. *Readings in Information Visualization*. Morgan Kaufmann, San Francisco, Calif., 1999.

[CNR96] N. Cooke, K. Neville, and A. Rowe. Procedural network representations of sequential data. *HumanComputer Interaction*, 11(1):2968, 1996.

[CRM91] S. Card, G. Robertson, and J. Mackinlay. The information visualizer, an information workspace. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 181–188, 1991.

[Dau92] I. Daubechies. *Ten Lectures on Wavelets*. SIAM, Phil-adelphia, Pennsylvania, 1992.

[Dav83] M. Davison. *Mutidimensional scaling*. John Wiley & Sons, 1983.

[DCHW03] M. Derthick, M. Christel, A. Hauptmann, and H. Wactlar. Constant density displays using diversity sampling. *Proc. IEEE Symposium on Information Visualization*, pages 137–144, 2003.

[DE01] R. Dachselt and J. Ebert. Collapsible cylindrical trees: A fast hierarchical navigation technique. *Proc. IEEE Symposium on Information Visualization*, pages 79–86, 2001.

[DE02] A. Dix and G. Ellis. By chance - enhancing interaction with large data sets through statistical sampling. *Proc. Advanced Visual Interfaces,* pages 167–176, 2002.

[Def77] D. Defays. An efficient algorithm for a complete link method. *Computer Journal*, 20(4):364–366, Nov. 1977.

[DHMR99] M. Derthick, J. Harrison, A. Moore, and S. Roth. Efficient multi-object dynamic query histograms. *Proc. IEEE Symposium on Information Visualization*, pages 58–64, October 1999.

[DPS96] K. Doan, C. Plaisant, and B. Shneiderman. Query previews in networked information systems. *ADL*, pages 120–129, 1996.

[DRRW03] P. Doshi, G. Rosario, E. Rundensteiner, and M. Ward. A strategy selection framework for adaptive prefetching in data visualization. *15th International Conference on Scientific and Statistical Database Management (SSDBM)*, pages 107–116, 2003.

[DRW03] P. Doshi, E. Rundensteiner, and M. Ward. Prefetching for visual data exploration. *Database Systems for Advanced Applications (DASFAA),* pages 195–202, 2003.

[Ead92] P. Eades. Drawing the trees. *Bulletin of the Institute of Combinatorics and its Applications*, pages 10–36, 1992.

[Eck21] M. Eckert. *Die Kartenwissenschaft, Forschungen und Grundlagen zu einer Kartographie als Wissenschaft*. Berlin und Leipzig, 1921.

[ESBB98] M. Eisen, P. Spellman, P. Brown, and D. Bostein. Cluster analysis and display of genome-wide expression patterns. *Proc. Natl. Acad. Sci. USA*, 95:14863–14868, 1998.

[Fle99] A. Flexer. On the use of self-organizing maps for clustering and visualization. *Proc. PKDD*, pages 80–88, 1999.

[For65] E. Forgy. Cluster analysis of multivariate data: Efficiency versus interpretability of classification. *Biometrics*, 21(3):768–780, 65.

[FR91] T. Fruchterman and E. Reingold. Graph drawing by force-directed placement. *Software - Practice and Experience*, 22(11):1129–1164, 1991.

[Fua99] Y. Fua. Hierarchical parallel coordinates. Master's thesis, Worcester Polytechnic Institute, 1999.

[Fur86] G. Furnas. Generalized fisheye views. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems,* pages 16–23, 1986.

[FWR99a] Y. Fua, M. Ward, and E. Rundensteiner. Navigating hierarchies with structure-based brushes. *Proc. IEEE Symposium on Information Visualization*, pages 58–64, 1999.

[FWR99b] Y. Fua, M. Ward, and E. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. *Proc. IEEE Visualization*, pages 43–50, Oct. 1999.

[FZ98] G. Furnas and X. Zhang. MuSE: A multiscale editor. *ACM Symposium on User Interface Software and Technology*, page 107116, 1998.

[GCB$^+$97] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. *J. Data Mining and Knowledge Discovery*, 1(1):29–53, 1997.

[GKN04] E. Gansner, Y. Koren, and S. North. Topological fisheye views for visualizing large graphs. *Proc. IEEE Symposium on Information Visualization*, pages 175–182, 2004.

[GKS$^+$04] M. Granitzer, W. Kienreich, V. Sabol, K. Andrews, and W. Klieber. Evaluating a system for interactive exploration of large, hierarchically structured document repositories. *Proc. IEEE Symposium on Information Visualization*, pages 127–134, 2004.

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

[GST$^+$01] J. Gray, A. Szalay, A. Thakar, P. Zunszt, T. Malik, J. Raddick, C. Stoughton, and J. vandenBerg. The SDSS SkyServer - public access to the Sloan Digital Sky Server data. Technical Report MSR-TR-2001-104, Microsoft, 2001.

[HB98] L. Holmquist and S. Björk. A hierarchical focus + context method for image browsing. *SIGGRAPH 98 Sketches and Applications*, page 282, 1998.

[HB99] S. Hettich and S. Bay. The uci kdd archive [http://kdd.ics.uci.edu]. *Irvine, CA: University of California, Department of Information and Computer Science*, 1999.

[HBLH94] W. Hersh, C. Buckley, T. Leone, and D. Hickman. Ohsumed: An interactive retrieval evaluation and new large text collection for research. *Proc. ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 192–201, 1994.

[HBP02] K. Hornbæk, B. Bederson, and C. Plaisant. Navigation patterns and usability of zoomable user interfaces with and without an overview. *ACM Trans. Comput.-Hum. Interact.*, 9(4):362–389, 2002.

[HCS98] K. Hua, Y. Cai, and S. Sheu. Patching: A multicast technique for true video-on-demand services. *ACM Multimedia*, pages 191–200, 1998.

[HF01] K. Hornbæk and E. Frøkjær. Reading of electronic documents: the usability of linear, fisheye, and overview+detail interfaces. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 293–300, 2001.

[HHN00] S. Havre, E. Hetzler, and L. Nowell. Themeriver: Visualizing theme changes over time. *Proc. IEEE Symposium on Information Visualization*, pages 115–124, 2000.

[HK98] A. Hinneburg and D. Keim. An efficient approach to clustering in large multimedia databases with noise. *Proc. 4th ACM SIGKDD*, pages 58–65, 1998.

[HK04] P. Haas and C. König. A bi-level bernoulli scheme for database sampling. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 275–286, 2004.

[HLD02] H. Hauser, F. Ledermann, and H. Doleisch. Angular brushing of extended parallel coordinates. *Proc. IEEE Symposium on Information Visualization*, pages 127–130, 2002.

[HMM$^+$99] I. Herman, M. Marshall, G. Melancon, D. Duke, M. Delest, and J.-P. Domenger. Skeletal images as visual cues in graph visualization. *Eurographics/IEEE TCVG Symposium on Visualization*, pages 13–22, 1999.

[Hol97] L. Holmquist. Focus+context visualization with flip zooming and the zoom browser. *Extended Abstracts of Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 263–264, 1997.

[Hop96] H. Hoppe. Progressive meshes. *SIGGRAPH*, pages 99–108, 1996.

[ID90] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multidimensional geometry. *Proc. IEEE Visualization*, pages 361–378, 1990.

[IH00] T. Igarashi and K. Hinckley. Speed-dependent automatic zooming for browsing large documents. *ACM Symposium on User Interface Software and Technology*, pages 139–148, 2000.

[IP95] Y. Ioannidis and V. Poosala. Balancing histogram optimality and practicality for query result size estimation. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 233–244, 1995.

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

[JKM$^+$98] H. Jagadish, N. Koudas, S. Muthukrishnan, V. Poosala, K. Sevcik, and T. Suel. Optimal histograms with quality guarantees. *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 275–286, 1998.

[Jol86] J. Jolliffe. *Principal Component Analysis*. Springer Verlag, 1986.

[JS91] B. Johnson and B. Shneiderman. Tree maps: A space-filling approach to the visualization of hierarchical information structures. *Proc. IEEE Visualization*, pages 284–291, 1991.

[Kas98] S. Kaski. Dimensionality reduction by random mapping: Fast similarity computation for clustering. *Proc. IJCNN'98, International Joint Conference on Neural Networks*, 1:413–418, 1998.

[KBG02] Z. Karni, A. Bogomjakov, and C. Gotsman. Efficient compression and rendering of multi-resolution meshes. *Proc. IEEE Visualization*, pages 347–354, 2002.

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

[KGKB03] G. Kollios, D. Gunopulos, N. Koudas, and S. Berchtold. Efficient biased sampling for approximate clustering and outlier detection in large data sets. *IEEE Transactions on Knowledge and Data Engineering*, 15(5):1170–1187, 2003.

[KH81] B. Kleiner and J. Hartigan. Representing points in many dimensions by trees and castles. *Journal of the American Statistical Associatioin*, pages 260–272, 1981.

[KH98] D. Keim and A. Hermann. The gridfit algorithm: an efficient and effective approach to visualizing large amounts of spatial data. *Proc. IEEE Visualization*, pages 181–188, 1998.

[KKA95] D. Keim, H.-P. Kriegel, and M. Ankerst. Recursive pattern: a technique for visualizing very large amounts of data. *Proc. IEEE Visualization '95*, pages 279–286, 1995.

[KLS00] M. Kreuseler, N. Lopez, and H. Schumann. A scalable framework for information visualization. *Proc. IEEE Symposium on Information Visualization*, pages 27–36, 2000.

[Koh95] T. Kohonen. *Self Organizing Maps*. Springer Verlag, 1995.

[KR90] L. Kaufman and P. Rousseeuw. *Finding Groups in Data: An Introduction to Cluster Analysis*. John Wiley and Sons, 1990.

[Krü98] A. Krüger. Automatic graphical abstraction in intent-based 3d-illustrations. *AVI*, pages 47–56, 1998.

[KSSB90] H.-P. Kriegel, B. Seeger, R. Schneider, and N. Beckmann. The r*-tree: an efficient access method for geographic information systems. *Proceedings International Conference on Geographic Information Systems*, 1990.

[KvdWvW01] E. Kleiberg, H. van de Wetering, and J. van Wijk. Botanical visualization of huge hierarchies. *Proc. IEEE Symposium on Information Visualization*, pages 87–94, 2001.

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

[KW99] A. Christian K'́onig and G. Weikum. Combining histograms and parametric curve fitting for feedback-driven query result-size estimation. *Proceedings of the 25th International Conference on Very Large Data Bases*, pages 423–434, 1999.

[KY93] H. Koide and H. Yoshihara. Fractal approaches for visualizing huge hierarchies. *Proc. IEEE Symposium on Visual Languages*, pages 55–60, 1993.

[LA00] A. Leuski and J. Allan. Lighthouse: Showing the way to relevant information. *Proc. IEEE Symposium on Information Visualization*, pages 125–129, 2000.

[LLG99] K. Law, J. Lui, and L. Golubchik. Efficient support for interactive service in multi-resolution vod systems. *VLDB Journal*, 8(2):133–153, 1999.

[LR96] J. Lamping and R. Rao. The hyperbolic browser: A focus+context technique for visualizing large hierarchies. *Journal of Visual Languages and Computing*, 7(1):33–55, 1996.

[LRP95] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 401–408, 1995.

[LS02] X. Li and H. Shen. Time-critical multiresolution volume rendering using 3d texture mapping hardware. *Proc. IEEE symposium on Volume Visualization and Graphics*, pages 29–36, 2002.

[LWW90] J. LeBlanc, M. Ward, and N. Wittels. Exploring n-dimensional databases. *Proc. IEEE Visualization*, pages 230–237, 1990.

[MB88] G. McLachlan and K. Basford. *Mixture Models: Inference and Applications to Clustering*. Marcel Dekker, New York, NY, USA, 1988.

[MB95] T. Munzner and P. Burchard. Visualizing the structure of the world wide web in 3d hyperbolic space. *Proc. VRML*, pages 33–38, 1995.

[MD88] M. Muralikrishna and D. DeWitt. Equi-depth histograms for estimating selectivity factors for multi-dimensional queries. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 28–36, 1988.

[Mea92] A. Mead. Review of the development of multidimensional scaling methods. *The Statistician*, 33:27–35, 1992.

[MPM90] J. McDonald, K. Paap, and D. McDonald. Hypertext perspectives: Using pathfinder to build hypertext systems. *Pathfinder Associative Networks: Studies in Knowledge Organization*, page 197212, 1990.

[MRC91] J. Mackinlay, G. Robertson, and S. Card. The perspective wall: detail and context smoothly integrated. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 173–179, 1991.

[Mun97] T. Munzner. H3: Laying out large directed graphs in 3d hyperbolic space. *Proc. IEEE Symposium on Information Visualization*, pages 2–10, 1997.

[Mun98] T. Munzner. Drawing large graphs with h3viewer and site manager. *Proc. Graph Drawing ’98, number 1547 in Lecture Notes in Computer Science*, pages 384–393, 1998.

[MWBF98] N. Miller, P. Wong, M. Brewster, and H. Foote. Topic islands: A wavelet-based text visualization system. *Proc. IEEE Visualization*, pages 189–196, 1998.

[MZS99] S. Mustière, J.-D. Zucker, and L. Saitta. Cartographic generalization as a combination of representing and abstracting knowledge. *ACM-GIS*, pages 162–164, 1999.

[NH94] R. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. *VLDB’94*, pages 144–155, 1994.

[NHT01] L. Nowell, E. Hetzler, and T. Tanasse. Change blindness in information visualization: A case study. *Proc. IEEE Symposium on Information Visualization*, pages 15–22, 2001.

[Nie92] J. Nielsen. Finding usability problems through heuristic evaluation. *Human Factors in Computing Systems, CHI'92 Conference Proceedings, p.372-380*, 1992.

[Nie94] J. Nielsen. Enhancing the explanatory power of usability heuristics. *Human Factors in Computing Systems, CHI'94 Conference Proceedings, p.152-158*, 1994.

[Nie95] J. Nielsen. keynote. *The IFIP INTERACT'95 International Conference on Human-Computer Interaction*, 1995.

[NIS05] Nist/sematech e-handbook of statistical methods. http://www.itl.nist.gov/div898/handbook/eda/section3/histogra.htm, 2005.

[NJS98] G. Nielson, I.-H. Jung, and J. Sung. Wavelets over curvilinear grids. *Proc. IEEE Visualization*, pages 313–317, 1998.

[NL94] J. Nielsen and J. Levy. Measuring usability: Preference vs. performance. *Commun. ACM*, 37(4):66–75, 1994.

[NM90] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. *Human Factors in Computing Systems, CHI'90 Conference Proceedings, p.249-256*, 1990.

[NS00] C. North and B. Shneiderman. Snap-together visualization: A user interface for coodinating visualizations via relational schemata. *Advanced Visual Interfaces*, pages 128–135, 2000.

[NSP96] C. North, B. Shneiderman, and C. Plaisant. User controlled overviews of an image library: A case study of the visible human. *Digital Libraries*, pages 74–82, 1996.

[Ohm94] J.-R. Ohm. Three-dimensional subband coding with motion compensation. *IEEE Trans. Image Processing*, 3(5):559–571, 1994.

[OO02] C. Ordonez and E. Omiecinski. Frem: fast and robust em clustering for large data sets. *CIKM*, pages 590–599, 2002.

[PCS95] C. Plaisant, D. Carr, and B. Shneiderman. Image browsers: Taxonomy, guidelines, and informal specifications. *IEEE Softw.*, 12(2):2132, 1995.

[PGB02] C. Plaisant, J. Grosjean, and B. Bederson. Spacetree: Supporting exploration in large node link tree, design evolution and empirical evaluation. *Proc. IEEE Symposium on Information Visualization*, pages 57–64, 2002.

[PHIS96] V. Poosala, P. Haas, Y. Ioannidis, and E. Shekita. Improved histograms for selectivity estimation of range predicates. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 294–305, 1996.

[PS97] E. Pedersen and T. Sokoler. Aroma: Abstract representation of presence supporting mutual awareness. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 51–58, 1997.

[PSC84] G. Piatetsky-Shapiro and C. Connell. Accurate estimation of the number of tuples satisfying a condition. *Proc. ACM SIGMOD International Conference on Management of Data*, pages 256–276, 1984.

[Qua01] J. Quackenbush. Computational analysis of microarray data. *Nature Reviews*, 2:418–427, 2001.

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

[RC94] R. Rao and S. Card. The table lens: merging graphical and symbolic representations in an interactive focus+context visualization for tabular information. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 318–322, 1994.

[RC95] R. Rao and S.K. Card. Exploring large tables with the table lens. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 403–404, 1995.

[RM93] G. Robertson and J. Mackinlay. The document lens. *Proc. ACM Symposium on User Interface Software and Technology*, pages 101–108, 1993.

[RMC91] G. Robertson, J. Mackinlay, and S. Card. Cone trees: Animated 3d visualization of hierarchical information. *Proc. ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 189–194, 1991.

[SCGM00] J. Stasko, R. Catrambone, M. Guzdial, and K. Mcdonald. An evaluation of space-filling information visualizations for depicting hierarchical structures. *Int. J. Human-Computer Studies*, 53:663–694, 2000.

[SFGF72] J. Siegel, E. Farrell, R. Goldwyn, and H. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery*, 72:126–141, 1972.

[SH00] C. Stolte and P. Hanrahan. Polaris: A system for query, analysis, and visualization of multidimensional relational databases. *Proc. IEEE Symposium on Information Visualization*, pages 5–14, 2000.

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

[Shn94] B. Shneiderman. Dynamic queries for visual information seeking. *IEEE Software*, 11(6):7077, 1994.

[Shn97] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley Publishing, 3rd edition, 1997.

[Sib73] R. Sibson. Slink: an optimally efficient algorithm for the single-link cluster method. *Computer Journal, vol.16(1)*, 16(1):30–34, Feb. 1973.

[SND04] P. Saraiya, C. North, and K. Duca. An evaluation of microarray visualization tools for biological insight. *Proc. IEEE Symposium on Information Visualization*, pages 1–8, 2004.

[Spe01] R. Spence. *Information Visualisation*. Addison-Wesley, 2001.

[SRW00] I. Stroe, E. Rundensteiner, and M. Ward. Scalable visual hierarchy exploration. *Database and Expert Systems Applications (DEXA)*, pages 784–793, September 2000.

[SS77] J. Smith and D. Smith. Database abstractions: Aggregation and generalization. *Commun. ACM*, 20(6):405–413, 1977.

[SS02] J. Seo and B. Shneiderman. Interactively exploring hierarchical clustering results. *Computer*, 35(7):80–86, 2002.

[SSTR93] M. Sarkar, S. Snibbe, O. Tversky, and S. Reiss. Stretching the rubber sheet: a metaphor for viewing large layouts on small screens. *Proc. UIST*, pages 81–91, 1993.

[STH02] C. Stolte, D. Tang, and P. Hanrahan. Multiscale visualization using data cubes. *Proc. IEEE Symposium on Information Visualization*, pages 1–8, October 2002.

[Str00] I. Stroe. Scalable visual hierarchy exploration. Master's thesis, Worcester Polytechnic Institute, May 2000.

[SW01] B. Shneiderman and M. Wattenberg. Ordered treemap layouts. *Proc. IEEE Symposium on Information Visualization*, pages 73–78, 2001.

[SZ98] L. Saitta and J.-D. Zucker. Semantic abstraction for concept representation and learning. *Proc. Symposium on Abstraction, Reformulation and Approximation*, 1998.

[SZ00] J. Stasko and E. Zhang. Focus+context display and navigation techniques for enhancing radial, space-filling hierarchy visualization. *Proc. IEEE Symposium on Information Visualization*, pages 57–65, 2000.

[TBS97] E. Tanin, R. Beigel, and B. Shneiderman. Design and evaluation of incremental data structures and algorithms for dynamic query interfaces. *Proc. IEEE Symposium on Information Visualization*, pages 81–86, 1997.

[Tho92] S. Thompson. *Sampling*. John Wiley and Sons, Inc., 2th edition, 1992.

[TLH$^+$00] E. Tanin, A. Lotem, I. Haddadin, B. Shneiderman, C. Plaisant, and L. Slaughter. Facilitating data exploration with query previews: A study of user performance and preference. *Behaviour & Information Technology*, 19(6):393–403, 2000.

[Tob70] W. Tobler. A computer model simulating urban growth in the detroit region. *Economic Geography*, 46(2):234–240, 1970.

[TZ94] D. Taubman and A. Zakhor. Multirate 3-d subband coding of video. *IEEE Trans. Image Processing*, 3(5):572–588, 1994.

[vdPvS00] P. van der Putten and M. van Someren. Coil challenge 2000: The insurance company case. Technical Report Technical Report 2000-09, Leiden Institute of Advanced Computer Science, 2000.

[vHvdWvW01] F. van Ham, H. van de Wetering, and J. van Wijk. Visualization of state transition graphs. *Proc. IEEE Symposium on Information Visualization*, pages 59–66, 2001.

[War00] C. Ware. *Information Visualization: Perception for Design*. Harcourt Publishers Ltd, 2000.

[War02] M. Ward. A taxonomy of glyph placement strategies for multidimensional data visualization. *Information Visualization*, 1(3-4):194–210, 2002.

[WB96] P. Wong and R. Bergeron. Multi-resolution multidimensional wavelet brushing. *Proc. IEEE Visualization*, pages 141–148, 1996.

[Weg90] E. Wegman. Hyperdimensional data analysis using parallel coordinates. *Journal of the American Statistical Association*, 411(85):664–675, 1990.

[WFA$^+$03] P. Wong, H. Foote, D. Adams, W. Cowley, and J. Thomas. Dynamic visualization of transient data streams. *Proc. IEEE Symposium on Information Visualization*, pages 97–104, 2003.

[Wil96] G. Wills. Selection:524,288 ways to say this is interesting. *Proc. IEEE Symposium on Information Visualization*, pages 54–59, 1996.

[WLS98] A. Woodruff, J. Landay, and M. Stonebraker. Goal-directed zoom. *Summary of the ACM Conference on Human Factors in Computing Systems*, pages 305–306, 1998.

[WLT94] M. Ward, J. LeBlanc, and R. Tipnis. N-land: a graphical tool for exploring n-dimensional data. *Proc. Computer Graphics International Conference*, pages 130–141, 1994.

[WS92] C. Williamson and B. Shneiderman. The dynamic home-finder: Evaluating dynamic queries in a real-estate information exploration system. *Proc. ACM SIGIR 92*, page 339346, 1992.

[WS99] J. Van Wijk and E. Van Selow. Cluster and calendar based visualization of time series data. *Proc. IEEE Symposium on Information Visualization*, pages 4–9, 1999.

[WS03] H. Wang and K. Sevcik. A multi-dimensional histogram for selectivity estimation and fast approximate query answering. *Proceedings of the 2003 conference of the Centre for Advanced Studies conference on Collaborative research*, pages 328–342, 2003.

[WTP$^+$95] J. Wise, J. Thomas, K. Pennock, D. Lantrip, M. Pottier, A. Schur, and V. Crow. Visualizing the non-visual: Spatial analysis and interaction with information from text documents. *Proc. IEEE Symposium on Information Visualization*, pages 51–58, 1995.

[WY04] M. Ward and J. Yang. Interaction spaces in data and information visualization. *Eurographics/IEEE TCVG Symposium on Visualization*, pages 137–146, 2004.

[XEKS98] X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. *Proceedings of the 14th ICDE*, pages 324–331, 1998.

[XMD] Xmdvtool home page. http://davis.wpi.edu/ xmdv/. http://davis.wpi.edu/˜xmdv.

[YFDH01] K.-P. Yee, D. Fisher, R. Dhamija, and M. Hearst. Animated exploration of graphs with radial layout. *Proc. IEEE Symposium on Information Visualization*, pages 43–50, 2001.

[YPH$^+$04] J. Yang, A. Patro, S. Huang, N. Mehta, M. Ward, and E. Rundensteiner. Value and relation display for interactive exploration of high dimensional datasets. *Proc. IEEE Symposium on Information Visualization*, pages 73–80, 2004.

[YPWR03] J. Yang, W. Peng, M. Ward, and E. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. *Proc. IEEE Symposium on Information Visualization*, pages 105–112, 2003.

[YWR02] J. Yang, M. Ward, and E. Rundensteiner. InterRing: An interactive tool for visually navigating and manipulating hierarchical structures. *Proc. IEEE Symposium on Information Visualization*, pages 77–84, 2002.

[YWR03a] J. Yang, M. Ward, and E. Rundensteiner. Hierarchical exploration of large multivariate data sets. *Data Visualization: The State of the Art 2003*, pages 201–212, 2003.

[YWR03b] J. Yang, M. Ward, and E. Rundensteiner. Interactive hierarchical displays: A general framework for visualization and exploration of large multivariate data sets. *Computers & Graphics*, 27(2):265–283, 2003.

[YWRH03] J. Yang, M. Ward, E. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. *Eurographics/IEEE TCVG Symposium on Visualization*, pages 19–28, 2003.

[YWRP03] J. Yang, M. Ward, E. Rundensteiner, and A. Patro. Interring: a visual interface for navigating and manipulating hierarchies. *Information Visualization*, 2(1):16–30, 2003.

[ZRL96] T. Zhang, R. Ramakrishnan, and M. Livny. Birch: an efficient data clustering method for very large databases. *SIGMOD Record*, 25(2):103–114, June 1996.