Interactive Hierarchical Displays: A General Framework for Visualization and Exploration of Large Multivariate Data Sets

Jing Yang, Matthew O. Ward [1] , and Elke A. Rundensteiner

Computer Science Department

Worcester Polytechnic Institute

Worcester, MA 01609

Abstract

Numerous multivariate visualization techniques and systems have been developed in the past three decades to visually analyze and explore multivariate data being produced daily in application areas ranging from stock markets to the earth and space spaces. However, traditional multivariate visualization techniques typically do not scale well to large multivariate data sets, with the latter becoming more and more common nowadays. This paper proposes a general framework for interactive hierarchical displays (IHDs) to tackle the clutter problem faced by traditional multivariate visualization techniques when analyzing large data sets. The underlying principle of this framework is to develop a multi-resolution view of the data via hierarchical clustering, and to use hierarchical variations of traditional multivariate visualization techniques to convey aggregation information about the resulting clusters. Users can then explore their desired focus region at different levels of detail, using our suite of navigation and filtering tools. We describe this IHD framework and its full implementation on four traditional multivariate visualization techniques,

namely, parallel coordinates [9,19], star glyphs [16], scatterplot matrices [4], and dimensional stacking [12], as implemented in the XmdvTool system [18,13,6,7]. We also describe an empirical evaluation that verified the effectiveness of the interactive hierarchical displays.

Key words: Large-scale multivariate data visualization, exploratory data analysis, hierarchical data exploration

---

1 Introduction

One important approach to supporting the human in analyzing and exploring large amounts of data is to graphically present the data and then allow the human to apply his or her perceptual abilities to make sense of the data. Multivariate visualization is an important subfield of data visualization that focuses on data items composed of more than two variables. Many multivariate visualization techniques and systems have emerged during the last three decades, such as glyph techniques [2,16,3,14], parallel coordinates [9,19], scatterplot matrices [4], pixel-level visualization [11], and dimensional stacking [12]. Each method has strengths and weaknesses in terms of the data characteristics and analysis tasks for which it is best suited.

As large data sets become more and more common, it has become clear that most existing multivariate visualization techniques lose their effectiveness when more than a few hundred or thousand data points are being displayed. The reason is that the available screen space is limited. Hence when the size of a data set reaches a certain size, we are not able to place all data on the

* Author to whom correspondences should be directed

screen at the same time without completely cluttering the screen. 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 of the existing multivariate 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 large multivariate data sets. Figure 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 [9,19]. 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.

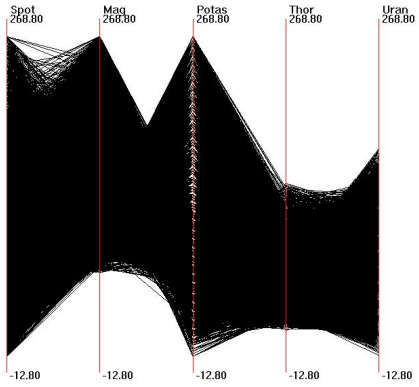

Fig. 1. The clutter problem. A remotely sensed data set, obtained from Peter Ketelaar, CSIRO Division of Exploration and Mining (size: 16384 data items, 5 dimensions) visualized with parallel coordinates.

To address the problem of scaling multivariate visualization techniques to large data sets, we have developed a general framework called *Interactive Hierarchical Displays* (IHDs). This framework is general in that it can be applied to a wide range of existing data visualization techniques. IHDs work upon hierarchical cluster trees of the data sets and display the data in variable levels of

detail. Each cluster is visualized using a technique called *the meanpoint band*, which conveys several attributes of the cluster. Hierarchical relationships are depicted using color, so that sibling and parent relations are readily observed. We call this *proximity-based coloring*, and it has proved to be powerful not only for conveying these relationships, but also for linking different visual representations of each cluster. Several tools to support interactive exploration within the IHD framework have also been developed, including techniques for intuitive navigation of the hierarchy via focus, drill-down, and roll-up operations, and selective filtering that reduces clutter while preserving context.

As a proof of the generality of IHDs, we have applied this framework to four diverse multivariate visualization techniques, namely parallel coordinates [9,19], star glyphs [16], scatterplot matrices [4], and dimensional stacking [12]. We have assessed these interactive hierarchical displays and found that they are more effective than the non-hierarchical techniques (which we call *flat* methods) in many aspects, such as for detecting outliers and linking individual data items within and between displays, as well as solving the clutter problem. We have developed a fully functioned system based on XmdvTool [18,13,6,7], a public-domain visualization system, that integrates hierarchical versions of the four supported techniques for displaying and visually exploring multivariate data. Application of these tools to real data sets has demonstrated the utility of the resulting technology.

To assess the effectiveness of the interactive hierarchical displays, we conducted an empirical evaluation of the hierarchical parallel coordinates and flat parallel coordinates. The results of this evaluation showed that the IHDs provided users with effective help in exploring large data sets as compared to traditional flat display techniques, and our interactive exploration tools were

useful and effective.

This paper is organized as follows: Section 2 gives an overview of the interactive hierarchical display framework. Section 3 introduces hierarchical versions of four traditional display techniques. Section 4 describes the full implementation of this framework in the XmdvTool system. Section 5 introduces the empirical evaluation. Section 6 surveys related work regarding the display of large multivariate data sets. We conclude with a summary of our contributions and open areas for future work in Section 7.

2 Interactive Hierarchical Display Framework

Our goal is to overcome the clutter problem faced by the traditional flat visualization techniques as illustrated in Section 1. Towards this end, we have developed novel interactive displays with the following properties:

- scale to large data sets while overcoming the clutter problem,
- give users information ranging from the high level structure of the data sets in the multidimensional space down to the detail of individual data items,
- follow a set of display-independent principles for all the display techniques to give users a consistent interpretation of displayed information across multiple views of the data.
- build off of existing visualization techniques so that users can readily interpret them,
- use a multiresolution approach that allows users to easily switch between different levels-of-detail,
- allow users to focus on particular subregions of the overall data sets and

view details while keeping the overall context, and

- preserve most, if not all, of the interactive techniques available for the traditional displays, such as brushing, zooming, and distortion.

As an initial step, we construct a hierarchical cluster tree upon a data set using a clustering algorithm, such as Birch [23], or via explicit partitioning. By cutting the hierarchical cluster tree at various depths, we get subsets of clusters at different levels-of-detail that abstract all data in the data set. Then, we can visualize the subsets of clusters instead of all data in the data set. This makes IHDs scalable to large data sets and able to provide multiple levels of information. In the remainder of this section we describe the general procedure for generating visualizations within the IHD framework and provide details on the interactive tools necessary for exploring data presented in this manner. Note that because of the generality of the visualization mappings, one set of interactive tools can be applied to a diverse set of hierarchical visualizations.

2.1 Visualization Within IHD

The following subsections describe the components of hierarchical visualization: the clustering process and methods for describing cluster contents, the visual representation of cluster descriptors, and the coloring scheme used to represent cluster relationships. We feel that conveying structural relationships between clusters is as important as providing users with the numeric contents of the clusters, and we will use this relational information as a mechanism to support navigation within a visualization as well as linkage between visualizations.

2.1.1 Hierarchical Cluster Tree

To overcome the problem of clutter on the display, the key strategy we employ is to put fewer items on the screen. Thus we need to compress the data sets while preserving their significant features. Moreover, we prefer multiresolution displays so that users can interactively select their preferred level of detail. Given the above considerations, the procedure we follow is to construct a hierarchical cluster tree for a data set and to provide visualization techniques designed to convey this tree.

A hierarchical cluster tree is typically formed by grouping objects based on some measure of proximity between pairs of objects [10]. A number of clustering algorithms have been proposed for building hierarchical cluster trees of large data sets [1,8,23]. Since our IHDs are independent of the particular choice of the clustering algorithms as long as it generates a hierarchical cluster tree, we will not discuss further the actual algorithm to construct such a tree. Rather we focus now on characterizing the properties of such a tree once constructed.

A hierarchical cluster tree is constructed upon a data set. Each node T_i of the hierarchical cluster tree presents a cluster. A non-leaf cluster is composed of all its child clusters, while a leaf cluster contains only a single data item. A hierarchical cluster tree structures and presents a large data set at different levels of abstraction. On extreme points, the collection of all leaf clusters presents exactly every data item of the data set, while the root is a cluster presenting the whole data set as one single node of the tree.

The following information may be directly obtained from a node T_i of the cluster tree:

- n_i : the number of data points enclosed, also called the population.
- m_i : the mean of the data points.
- e_i : the minimum and maximum bounds of the cluster for each dimension, also called the extents.
- v_i : a measure of the size of cluster T_i
- l_i : the tree depth or level at node T_i

Note that v_i is a computed measure of the cluster size and satisfies the following criteria: If T_i is an ancestor of T_j, then

$$v_i >= v_j.$$

To visualize a data set at a certain level of detail (LOD), we display a subset of clusters in its hierarchical cluster tree T on the screen, instead of all the data in the data set (i.e., all leaves of the tree). We use the following approach to get the subset of clusters. We first find the range of v within the data set:

$$v_{max} = \max_{T_i \in \mathbf{T}} \{v_i\}$$
$$v_{min} = \min_{T_i \in \mathbf{T}} \{v_i\}$$

Then we define the LOD control parameter $w \in [v_{min}, v_{max}]$. This then leads to the definition of a cut $S(w)$ of a tree at a given level of detail:

$$S(w) = \{ \ T_i \ | \ (\ v_i \leq w \text{ or } \ T_i \text{ is a leaf node }) \text{ and } v_{parent(i)} \geq w \}$$

Note that $S(v_{max})$ is a single cluster representing the whole data set, namely, the root, while $S(v_{min})$ is composed of all the leaf clusters representing every data item in the data set. $S(w)$ denotes the subset of clusters that we want to display on the screen. $S(w)$ changes smoothly with the LOD control parameter

w. $S(w)$ can be intuitively envisioned as a horizontal cut across the hierarchical cluster tree T that satisfies the following criteria: for each path from the root to a leaf, $S(w)$ intersects the path at exactly one point. The position of the cut changes when w changes. The higher w is, the closer the cut is to the root of the tree; the lower w is, the closer the cut is to the leaves of the tree.

The reason that we choose w as the LOD control parameter is that w is continuous and thus provides smooth transitions on our hierarchical display. An example of a discontinuous LOD control parameter is the tree depth, whose change can also cause $S(w)$ to change. However, it is a poor choice in some cases because the number of nodes may increase dramatically with depth. This would manifest itself as abrupt screen changes as the LOD switches values at higher depths of the tree.

As we will show later, with proper interactive tools to support drill-down/roll-up operations, users can switch among different levels of detail easily by changing the LOD control parameter w directly.

2.1.2 Meanpoint-Band Method

In our interactive hierarchical displays, we visualize subsets of clusters in the hierarchical cluster tree on the screen instead of all the data in the data set. Thus an importance issue is how to display the salient feature of clusters. This ideally should happen in a manner consistent with the traditional display technique already chosen by the user, so that the behavior of the hierarchical display is predictable and familiar to the user. For this purpose, we introduce the *meanpoint-band* method to draw clusters. This technique conveys several essential features of the clusters, such as the mean and extents, while preserv-

ing all characteristics of the existing traditional display techniques.

The *meanpoint* in our context refers to the mean of a cluster. It is assigned the color of its cluster, using the proximity-based coloring strategy described in Section 2.1.3, and is displayed like an ordinary data item in traditional (flat) displays. Surrounding a meanpoint we place a *band* that indicates the extent of the respective cluster in each dimension. Bands look different in different display techniques, but they always extend from the minimum to the maximum value in each dimension of the corresponding cluster and represent all data items in the clusters. A band is also assigned the color of the cluster it represents. To give the user a sense of the location of data points in a cluster and to convey the overlap among clusters, unlike a meanpoint, which is solid, a band is translucent. We assume that there is a linear drop-off in the density of cluster data from its center to the edge, and set the maximum opacity proportional to the population.

Figure 2 is an example of a two-dimensional cluster showed by the meanpoint-band method in a plot. The mean of this cluster is $(x0, y0)$, the cluster range in the X dimension is $[x2, x1]$ and the cluster range in the Y dimension is $[y2, y1]$. The distance from the meanpoint to the maximum value in the X dimension equals to $x1 - x0$, in Y dimension equals to $y1 - y0$. The distance from the meanpoint to the minimum value in X dimension equals to $x0 - x2$, in the Y dimension equals to $y0 - y2$. When we mention the extent of a cluster, we mean these distances.

The meanpoint-band method for data display has driven the development of many of our interactive tools, such as extent scaling (Section 2.2.3) and dynamic masking (Section 2.2.4). A key property of this solution is that it not

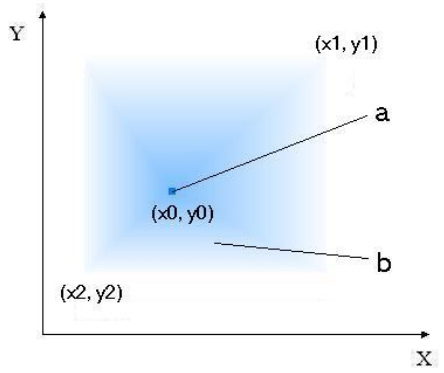

Fig. 2. A cluster shown by the meanpoint-band method in a 2-D plot. (a) The meanpoint of the cluster; (b) The band of the cluster.

only helps establish coherence between a traditional display technique and its hierarchical version, but also provides linkage between different hierarchical displays.

2.1.3 Proximity-Based Coloring

Cohesion (associating the components of a single data item) and linkage (associating multiple views of the same data item within a single visualization or between multiple visualizations) are common problems when analyzing multivariate data. Users may misinterpret data when examining a parallel coordinates display, as the polylines representing them intersect each other at one or more axes. It can also be difficult to track a data point across multiple plots within a scatterplot matrix. Finally, a growing number of visualization tools provide multiple displays of the same data, thus increasing the need to provide effective linkage between views. One reason for the above problems is that typically all data are displayed using the same color. Our solution to these problems is to use color to distinguish different data items and link the same data item in different views. Indeed, it is common to allow users to map one or more data dimensions to the color attribute. In the context

of our interactive hierarchical displays, we have chosen color to convey relational information, namely those represented by the hierarchy. This gives us the benefits of establishing cohesion and linkage as mentioned above, while at the same time emphasizing the structural relationships between clusters and establishing correspondences between the hierarchical cluster trees and the interactive hierarchical displays.

We introduce a coloring strategy, called *proximity-based coloring*, to assign colors to clusters in the hierarchical cluster tree. It maps colors by cluster proximity based on the structure of the hierarchical cluster tree. It has the following properties:

- sibling clusters have similar colors (thus the name proximity-based coloring) to depict the high proximity among the sibling clusters,
- a parent cluster has a color within the range of its children's colors to convey that the parent cluster is composed of its children clusters,
- the color space is effectively utilized, i.e., there are no significant parts of the color space to which no cluster is assigned, and
- difference in color between non-sibling clusters are readily discernible compared to the difference between siblings.

The process of proximity-based coloring is to impose a linear order on all the clusters in the hierarchical cluster tree and assign colors to each cluster by indexing into a linear colormap table. The following algorithm [5] is an example of the proximity-based coloring for a binary hierarchical cluster tree, which is the simplest case.

- $C()$: the hue component of an HSV colormap. Alternate color maps are possible as well.

- T_i : the node i of the hierarchical cluster tree.
- $C(T_i) \in [0,1]$: the color assigned to the node i of the hierarchical cluster tree.
- T_0 : the root of the tree.
- $C(T_0)$: the color of the root.

For a binary hierarchical cluster tree, we can assign colors to its nodes based on the following recursive formula:

$$\begin{aligned} C(T_0) &= 0.5 \\ C(T_i) &= C(parent(T_i)) + \frac{\pi(i)}{K^{l_i+1}} \end{aligned} \tag{1}$$

where K is the branching factor of the cluster tree, l_i is the tree depth at node T_i, and $\pi(i)$ is the sign function defined as:

$$\pi(i) = \begin{cases} +1 & \text{if i is odd} \\ -1 & \text{if i is even} \end{cases} \tag{2}$$

This equation does not differentiate between adjacent elements (with respect to the linear order) belonging to different subtrees. It is important to distinguish between such elements because these adjacent elements are deemed "significantly separated" according to our proximity measure. For this, we revise Equation (1) by introducing a "buffer" between subtrees. The buffer acts as an unused color interval between subtrees so that elements at the proximal ends of subtrees are not assigned colors that are indistinguishable. Clearly the buffer should be larger between large clusters and smaller otherwise.

Let b, where $b < 1$, be the desired buffer interval. Let the revised definition be:

$$C(T_i) = C(parent(T_i)) + \pi(i)\left(b^{l_i} + \frac{1}{K^{l_i+1}}\right) \quad (3)$$

Equation (3) achieves our desired purpose. We typically choose b to be small with values around 10^{-1}.

Discussion of more complex approaches to color assignment for non-binary trees can be found in [5].

2.2 Interaction Within IHD

Having introduced the notion of the hierarchical cluster tree, proximity-based coloring and the meanpoint-band display method, we now need to tackle the problem of how to give the users the power to interactively perform their desired tasks. We have developed several interactive tools, such as the structure-based brush, drill-down/roll-up operations, extent scaling, and dynamic masking, to interactively explore the hierarchical displays. These are described below.

2.2.1 Structure-based Brush

Users often need to explore a particular subspace of interest after obtaining an overview of the hierarchical structure. One way of achieving this is though *brushing*. Brushing is a *direct* and *data-driven* metaphor. It is an interactive process for selecting subsets of data or localizing a subspace within an N-dimensional space [13,22,18]. Many useful operations, such as highlighting, deleting, masking or aggregation, may be performed on elements that lie within the selected hierarchical subspace. Brushing has traditionally been

performed in either *screen space* or *data space*. One example of brushing in screen space is the use of rubber-banding rectangles; an example of brushing in data space is interactively creating hyperboxes by painting over data points of interest [13].

Since brushing has been shown to be useful, we want to develop ways to use it in interactive hierarchical displays. However, brushing in screen space or data space cannot perform necessary selection operations in our hierarchical displays. Because the hierarchical cluster tree is highly structured, we have developed the concept of a *structure-based brush*. A structure-based brush allows users to select subsets of a data structure by specifying focal regions as well as a levels-of-detail on a visual representation of the structure. Details of the structure-based brush can be found in [7,5].

Figure 3 depicts a structure-based brush for hierarchically structured data. The triangular frame represents the hierarchical cluster tree. The white polyline near the bottom of the display depicts the silhouette of the tree. It delineates the approximate shape formed by chaining the leaf nodes. The colored bold contour across the middle of the tree delineates the tree cut $S(w)$ that represents the cluster partition corresponding to a level-of-detail w (Section 2.1.1). The colors on the contour correspond to the colors used for drawing the nodes on the data display (Section 2.1.3). The two movable handles on the base of the triangle, together with the apex of the triangle, form a wedge in the hierarchical space.

Interaction with this brush entails localizing a subspace within the hierarchical space by positioning the two handles at the base of the triangle. The embedded wedge forms a brushed subspace within the hierarchical space. Elements within

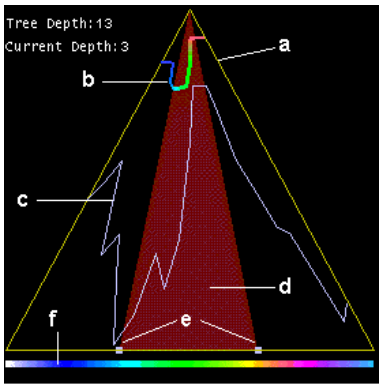

Fig. 3. Structure-based brushing tool. (a) Hierarchical cluster tree frame; (b) Contour corresponding to current level-of-detail; (c) Leaf contour approximates shape of hierarchical cluster tree; (d) Structure-based brush; (e) Interactive brush handles; (f) Colormap legend for level-of-detail contour.

the brushed subsets may be examined at different levels-of-detail (Section 2.2.2), magnified and examined in full view, or masked or emphasized using fading in/out operations (Section 2.2.4).

2.2.2 Drill-down/Roll-up Operations

Drill-down/roll-up operations allow users to change the level of detail of interactive hierarchical displays intuitively and directly. Drill-down refers to the process of viewing data at an increased level of detail, while roll-up refers to the process of viewing data with decreasing detail [6]. When users perform drill-down, the number of clusters in the display increases because accordant clusters split into smaller clusters. Conversely, when users perform roll-up, the number of clusters in the display decreases because the corresponding clusters merge together and form larger clusters.

As mentioned in Section 2.1.1, the drill-down/roll-up operations work directly by changing the LOD control parameter w, which causes $S(w)$, the subset of

the clusters displayed on the screen, to change in a smooth and continuous manner (Section 2.1.1). We couple our drilling operations with brushing. Our system permits selective drill-down/roll-up of the brushed and non-brushed region independently. This flexibility is important as it allows the viewing of a subset of elements in varying levels of detail while maintaining the overall context.

2.2.3 Extent Scaling

Though the bands of the clusters on the interactive hierarchical displays are translucent, it is often difficult to isolate or to tell them apart when they are overlapping. Moreover, it is possible that the users may want to see the bands indicating the relative scale of the clusters but do not want to see them covering a large portion of the screen.

One way to solve this problem is to decrease the extents of all the bands in each dimension by scaling them uniformly via a dynamically controlled extent scaling parameter $E \in [0, 1]$. E affects the extents of the bands in this way:

$$bandExtent_i = E * clusterExtent_i \tag{4}$$

where i refers to the identity of a cluster T_i, $clusterExtent_i$ refers to the distances from the mean to the maximum and minimum value of cluster T_i in each dimension, and $bandExtent_i$ refers to the distances from the meanpoint to the maximum and minimum value of the band of cluster i in each dimension.

Extent scaling can be illustrated using the example in Figure 4, which shows three two-dimensional clusters displayed using a scatterplot. The small circles are meanpoints of the clusters while the blocks are the bands of the clusters.

In the left plot, the bands of the clusters overlap. In the right plot, the overlaps are eliminated by extent scaling. Compared to the left plot, the meanpoints are not affected by the extent scaling, while the distance from the meanpoints to the maximum and minimum value of the bands have been proportionally reduced by E of 0.5.

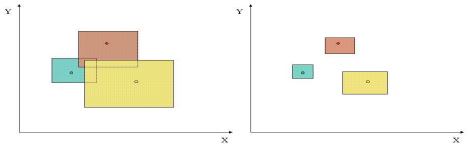

Fig. 4. Extent scaling. The left plot shows three two-dimensional clusters. The right plot shows the same clusters after extent scaling.

By reducing E, the overlaps of the bands are reduced and the screen space occupied by the bands shrunk. Though the extents of the bands no longer reflect the extents of the clusters when E is less than 1, they preserve the relative proportions among the different clusters. Thus the users can still differentiate between clusters with large and small extents after the bands have been scaled.

2.2.4 Dynamic Masking

Dynamic masking refers to the capability of controlling the relative opacity between brushed and unbrushed clusters (Section 2.2.1). It allows users to deemphasize or even eliminate brushed or unbrushed clusters. With dynamic masking, the viewer can interactively fade out the visual presentation of the unbrushed clusters, thereby obtaining a clearer view of the brushed clusters while maintaining context regarding unbrushed areas. Conversely, the bands

of the brushed clusters can be faded out, thus obtaining a clearer view of the unbrushed region. Used together with the structure-based brush, dynamic masking reduces the overlapping and density of the clusters on the screen by fading out uninteresting clusters so that users can concentrate on the clusters of interest.

Figure 5 is an example of dynamic masking. It shows three two-dimensional clusters displayed using scatter plots. The small circles are meanpoints of the clusters while the blocks are the bands of the clusters. In the left plot, the bands of the clusters overlap significantly. In the right plot, the yellow cluster is masked since the user does not want to study it at the present time. Notice that the band associated with the yellow cluster has been totally eliminated.

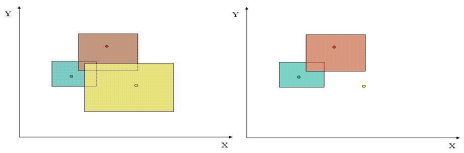

Fig. 5. Dynamic masking. The left plot shows three two-dimensional clusters. The right plot shows the same clusters after the yellow cluster is masked.

3 Four Hierarchical Display Techniques

We have extended four traditional multivariate visualization techniques, namely parallel coordinates, star glyphs, scatterplot matrices, and dimensional stacking, using the IHD framework. Below we outline how each of the four flat techniques can be generalized into a corresponding hierarchical display; the

key contribution is to convey the generality of our IHD concepts by showing that they are readily applicable to a large variety of visualization techniques.

3.1 Hierarchical Parallel Coordinates

In the basic form (flat) of parallel coordinates [9,19](Figure 6), each dimension is represented as a uniformly spaced vertical axis. A data item in this multidimensional space is mapped to a polyline that traverses across all the axes.

We generate hierarchical parallel coordinates from basic parallel coordinates using the meanpoint-band method. In the hierarchical parallel coordinates (Figure 7), the clusters replace the data items. The mean of a cluster is mapped to a polyline traversing across all the axes, with a band around it depicting the extents of the cluster in each dimension. The lower edge of the band intersects each axis at the minimum value of its respective cluster in that dimension. The upper edge of the band intersects each axis at the maximum value of its respective cluster in that dimension. Obviously, if we draw each data item included in that cluster, they will all be inside the band. Notice that even if two polylines intersect each other at some axis, we can easily differentiate them because they have different colors. Figure 8 improves upon Figure 7 by scaling the bands to reduce the overlaps among the clusters. Also, In Figure 8 we mask the brushed clusters so that users can concentrate on the unbrushed clusters. In Figure 9, the brushed region and unbrushed region are of different levels of detail. The brushed region is at such a low level of detail that it forms only two clusters with wide bands, while the unbrushed region is at a high level of detail, enabling all the individual data items to be seen.

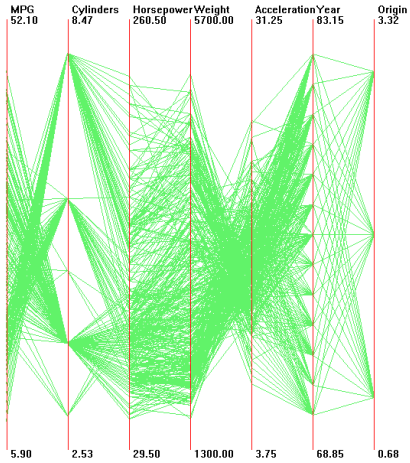

Fig. 6. Basic parallel coordinates.

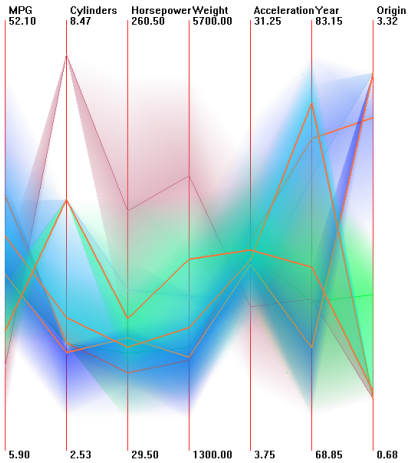

Fig. 7. Hierarchical parallel coordinates. The brushed region is highlighted.

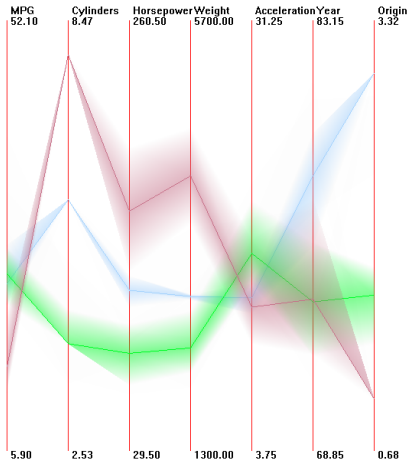

Fig. 8. Hierarchical parallel coordinates. The bands have been scaled to reduce overlapping. The brushed region has been masked.

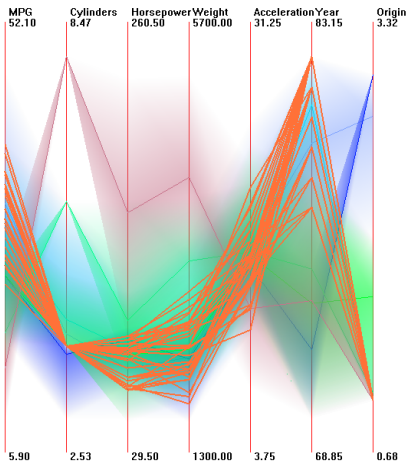

Fig. 9. Hierarchical parallel coordinates. The brushed region has a lower level of detail while the unbrushed region has a higher level of detail.

3.2 Hierarchical Glyphs

To visualize multivariate data with glyphs, each data point is represented by an individual shape [2,3,14]. 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(Figure 10). We can view these rays as axes, with each axis representing a dimension. The directions of these axes are from the center point to the outside.

Once again, we generated hierarchical star glyphs from the basic glyphs using the meanpoint-band method. In hierarchical glyphs (Figure 11), each star glyph represents a cluster. The mean values are used to generate the basic star shape. The band around the mean polygon has two edges. One is outside the mean polygon and another one is inside the mean polygon. The inside edge intersects each axis at the minimum value of its respective cluster in that dimension, while the outside edge intersects each axis at the maximum value of its respective cluster in that dimension. Obviously, if we draw a star glyph starting from the same center point to present a data item included in that cluster, this star glyph would be inside the band of that cluster. Thus the band successfully depicts the extent of the cluster.

Fig. 10. Basic star glyphs.

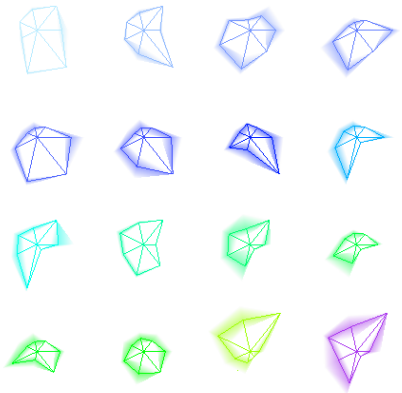

Fig. 11. Hierarchical star glyphs.

3.3 Hierarchical Scatterplot Matrix

In a scatterplot matrix (Figure 12), each data item is projected to $N*N$ plots, with N being the number of dimensions of the data set. The position of the projected point in a plot is decided by the values of the data item in the two dimensions that compose this plot.

We apply the meanpoint-band method to generate a hierarchical scatterplot matrix (Figure 13), by displaying the clusters in the $N*N$ plots. The mean of a cluster is drawn as an ordinary data item in the flat scatterplot matrix. The extents of each cluster forms rectangles around the projected mean in each plot. The projections of a cluster on different plots are drawn in the same color, which gives users the convenience of linking a cluster from one plot to another. In the flat form scatterplot matrix, all the data items have the same color. Hence users can have difficulty linking a data item when they move from one plot to another. Figure 14 improves upon Figure 13 by scaling the bands to reduce the overlaps. Also, the brushed clusters are masked. In Figure 15, the brushed region and unbrushed region are at different levels of detail, showing the terminal nodes of the unbrushed subtrees.

3.4 Hierarchical Dimensional Stacking

Dimensional stacking [12] (Figure 16) displays an N dimensional data set by recursively embedding pairs of dimensions within one another. Each dimension is descretized into a small number of subranges, and two dimensions are initially selected to subdivide the display space into subimages whose size and count depend on the number of subranges or bins used for those dimensions.

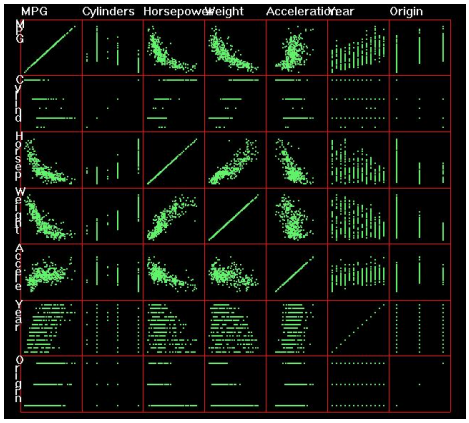

Fig. 12. Basic scatterplot matrix.

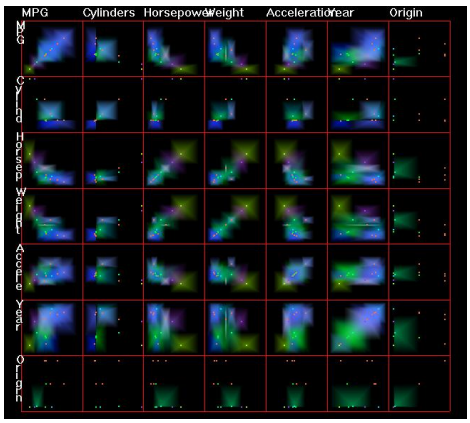

Fig. 13. Hierarchical scatterplot matrix.

Fig. 14. Hierarchical scatterplot matrix. The bands have been scaled to reduce the overlapping. The brushed clusters have been masked.

Fig. 15. Hierarchical scatterplot matrix. The brushed region has a lower level of detail while the unbrushed has a higher level of detail.

These subimages are subdivided based on the next two dimensions, and the process repeats until all dimensions have been mapped. Thus the multivariate space is split into a number of small cells that each map to a segment of the screen. Each data item will fall into one of these small cells, and thus have an assigned screen position.

In hierarchical dimensional stacking (Figure 17), the clusters replace the data items. The mean of a cluster will fall into a single small block in the same way as an ordinary data item in the flat form dimensional stacking. The band of this cluster depicting the cluster extents may potentially map to several bins. This time it is possible that some parts of the band are disjoint from others due to the embedding process, even though they are adjacent in N-dimensional space. Proximity-based coloring helps users to fuse the disjoint components of a particular cluster.

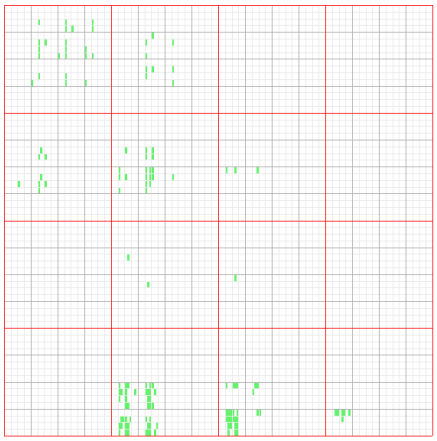

Fig. 16. Basic dimensional stacking.

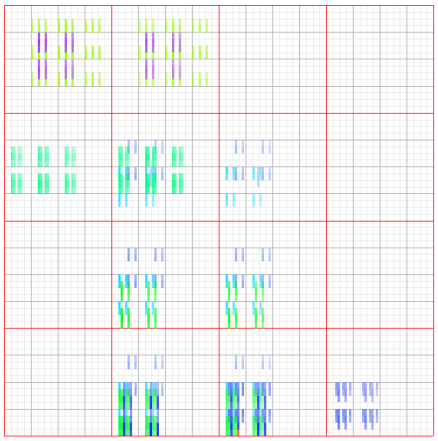

Fig. 17. Hierarchical dimensional stacking.

3.5 Towards More Hierarchical Display Techniques

The successful application of IHD principles to four rather diverse display techniques illustrates very clearly how other hierarchical display techniques can be generated from flat display techniques. It should be relatively straightforward to repeat the same procedure on other multivariate visualization techniques. Only the following steps are needed:

(1) Replace the data items in the flat form display technique by the clusters,
(2) Draw the means of the clusters using a similar method as drawing data items in the flat technique, with the addition of proximity-based coloring, and
(3) Draw opacity bands around the means to indicate the extents of the clusters.

4 Implementation

As a proof of concept, we have developed a full implementation of the IHDs as described in Section 2. In particular, we chose to build this prototype as an extension to the XmdvTool system (`http://davis.wpi.edu/~xmdv`), a freeware tool developed at WPI that incorporates four flat display techniques for visualizing and exploring multivariate data. Below we describe the basic architecture of the IHDs as implemented in XmdvTool4.1.

4.1 IHDs in XmdVTool4.1

Figure 18 illustrates how IHDs are implemented in XmdvTool4.1. It is composed of three modules, namely the hierarchical cluster tree module, the interactive navigation module, and the display module. The interactive navigation module is in charge of all the interactive tools. The display module is responsible for the display of all graphics on the screen. The hierarchical cluster tree module serves as a data center that connects the interactive navigation module and the display module by operating on a shared set of objects.

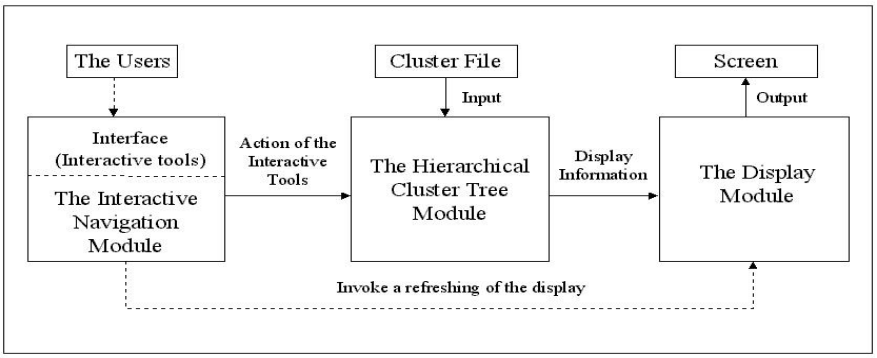

Fig. 18. The architecture of the IHDs in XmdvTool4.1

The hierarchical cluster tree module can be viewed as the data management center of the IHD framework. It is responsible for reading in the hierarchical cluster file, for constructing and managing the hierarchical cluster tree, for providing all information the display module needs to display graphics, and for maintaining the intermediate results of the interactive navigation module for further navigation.

The hierarchical cluster tree module constructs a hierarchical cluster tree by reading in a hierarchical cluster file. For very large data sets that can no longer be placed in main memory, the hierarchical cluster trees are kept in an Oracle database for persistent storage. A discussion of this database back-end is beyond the scope of this paper; interested readers are directed to [17].

Some display-oriented information comes directly from the cluster file, such as the index of a node's parent, the size, the extents of the cluster in all dimensions, and the mean of all data items included in each cluster in the hierarchical cluster tree. The hierarchical cluster tree module also stores additional information, such as the color, the on/off flag and the brushed/unbrushed flag for each node of the hierarchical cluster tree. These are initialized immediately after opening a new file and will be changed during interactive navigation. The on/off flag indicates if the cluster belongs to the current subset of clusters to be displayed on the screen. The brushed/unbrushed flag indicates if the cluster is currently selected by the active structure-based brush. The benefit of placing these flags with the hierarchical cluster tree is that the display module can read all the information needed for displaying directly from the

hierarchical cluster tree data structure when refreshing the screen. Otherwise the display module would have to regenerate them every time the screen needs to be refreshed. For example, if there were no brushed/unbrushed flags in the hierarchical cluster tree, the display module would have to recalculate which clusters are selected by the structure-based brush every time the screen needs to be refreshed, no matter if such a refresh was caused by the interactive navigation or other reasons. Since we store these flags in the hierarchical cluster tree, they need to be recalculated only when the brush is changed. The global parameters such as the dynamic masking parameter and the extent scaling parameter are also kept in this module. On one hand, they are global parameters that affect the display. On the other hand, they are intermediate results of the interactive navigation module to facilitate further navigation.

The hierarchical cluster tree module keeps track of the impact of any actions taken by the interactive tools. For example, the brushed/unbrushed flags of the nodes typically change after the user manipulates the structure-based brush, while the on/off flags of the nodes are refreshed every time the drill-down/roll-up operations cause the levels of detail to change. Similarly, the dynamic masking parameter and the extent scaling parameter always reflect the status of the dynamic masking tool and extent scaling tool respectively.

4.3 Interactive Navigation Module

The interactive navigation module encapsulates all the interactive tools mentioned in Section 2.2. After every manipulation of any of the interactive tools, the interactive navigation module checks if this action should have any effect on the display. If yes, it will inform the hierarchical cluster tree module. The

hierarchical cluster tree module then modifies the hierarchical cluster tree accordingly. After that, the interactive navigation module requests a refreshing of the display. This request causes the display module to read the updated information from the hierarchical cluster tree module and output updated graphics on the screen.

4.4 Display Module

The redraw of the output window will cause the display to be refreshed. In addition, the interactive navigation module can purposely invoke a refresh of the display. The task of the display module is to visually depict the currently selected subset of clusters on the screen using the active display technique every time the display needs to be refreshed. It gets all the needed information from the hierarchical cluster tree module. Since the hierarchical cluster tree module keeps track of any effect from the interactive tools, the display module needs not concern itself with the interactive tools at all.

The display module gets the selected subset of clusters by reading in all the nodes with "on" flags in the hierarchical cluster tree. The nodes with "off" flags are omitted since they are not in the selected subset to be displayed. The nodes with "unbrushed" marks are displayed using their assigned colors. The bands of the nodes with "brushed" marks are also displayed using their assigned colors, but their means are displayed using a special highlight color (red) to differentiate them from the unbrushed nodes.

The advantages of composing the IHDs via these three modules are obvious. First, the display module does not need to consider the actions of the interactive tools because the display module is separated from the interactive navigation module by the hierarchical cluster tree module. Second, changing the display technique will affect neither the interactive navigation module nor the hierarchical cluster tree module because they both have nothing to do with the display techniques. As a result, our system can be easily and flexibly extended to other display techniques besides the ones chosen for our current implementation.

5 Evaluation

The goal of our evaluation was to assess if our proposed framework of interactive hierarchical displays 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 our IHD framework, we selected one of them as a representative, namely, parallel coordinates, and then assessed it in both hierarchical and flat forms.

5.1 Experimental Methods and Setup

Our hypothesis was that, in general, subjects using the hierarchical parallel coordinates (HPC) could perform better in pattern finding (including clusters and outliers) of large data sets than subjects using the flat parallel coordinates (FPC). We expected a steeper learning curve of HPC than FPC since HPC is more complex than its flat counterpart. For one experiment, we simplified our test system 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.

5.1.1 Subjects

All the subjects in this evaluation were graduate students of the Worcester Polytechnic Institute. Two computer science major graduate students with data mining background attended our pre-experiments. The other subjects participated in either the HPC experiment or the FPC experiment (see Figure 19).

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

Fig. 19. Information about subjects in evaluation experiments. Prior background indicates how many had prior exposure to visualization or data mining.

5.1.2 Materials and Setup

To begin, a presentation was given in both the HPC and FPC experiments (this 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 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 dimensions and 150 data items. The AAUP data set contains 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 Pentium III PC. The patterns found by the subjects were saved into the local PC as pictures by the subjects themselves during the experiments and

collected together after the experiments.

5.1.3 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 starting 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.

5.2 Results and Discussion

Figure 20 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 all 25 patterns, which are all large clusters or obvious outliers.

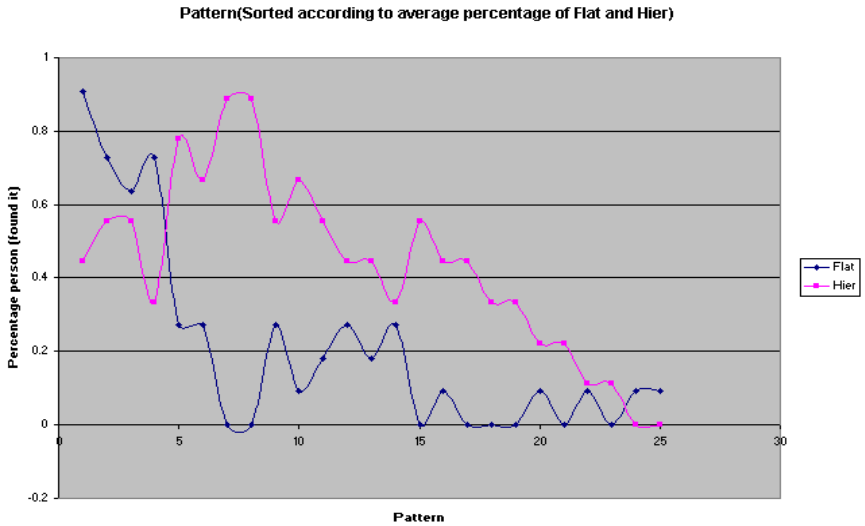

Fig. 20. Pattern Finding Results of the HPC and FPC Experiments

Figure 21 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.

Fig. 21. Average Time Interval in Pattern Finding

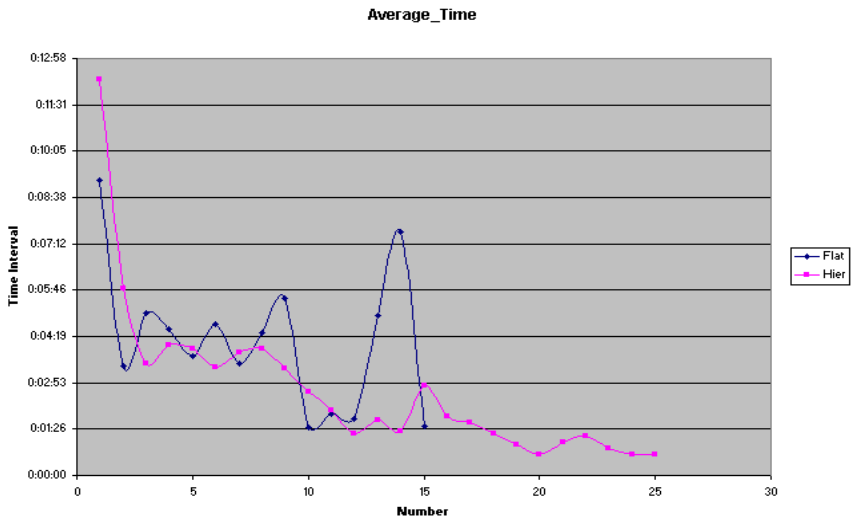

Figure 22 shows the individual subjects' process of pattern finding over time. The Y axis of the figures are the local times in the computers the subjects used. The X axis of the figures denotes the number of patterns saved by the subjects. In the left chart each line represents a subject in the FPC experiment, while in the right chart each line represents 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 with CS major and 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.

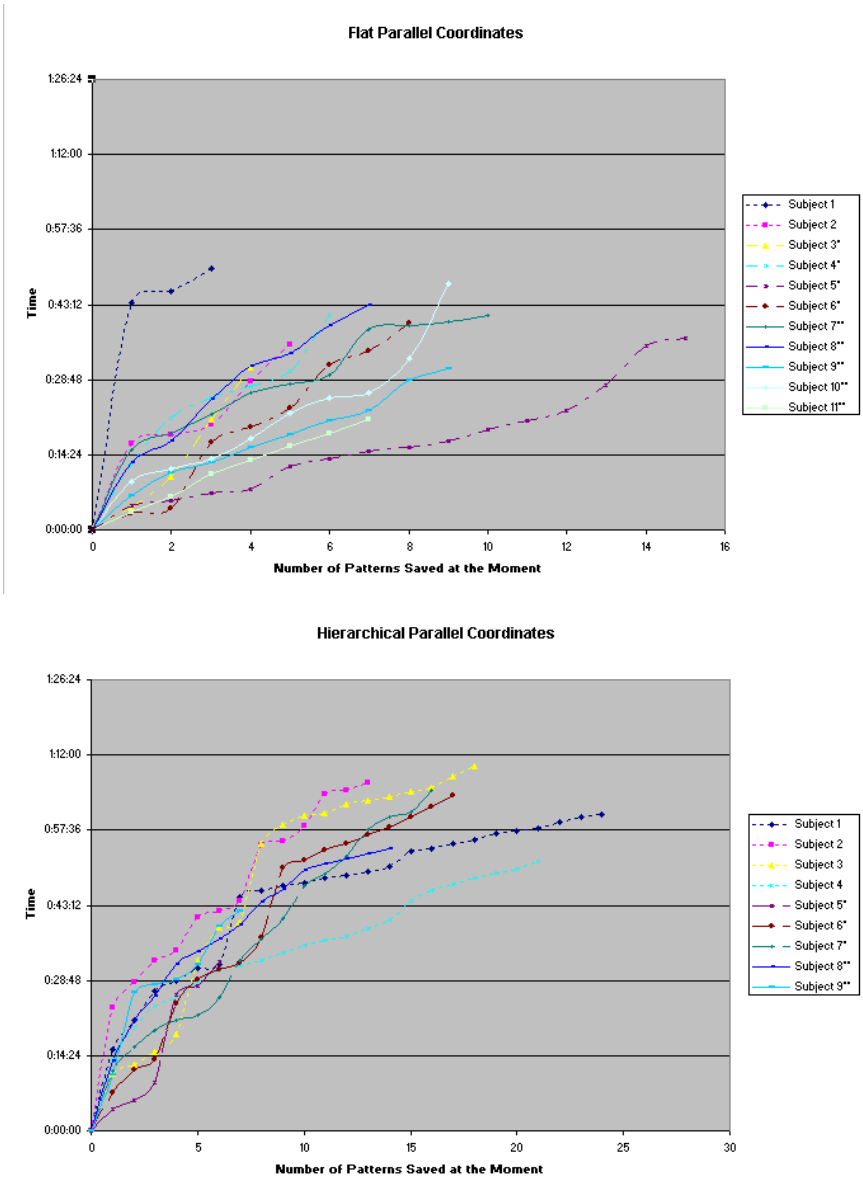

Fig. 22. Subjects' Process of Pattern Finding. **: CS student with relevant background; *: CS student; No Mark: non-CS student without relevant background

5.3 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 exactly.

6 Related Work

In recent years, many research efforts have focused on the problem of clutter when visualizing large multivariate data sets.

Wong and Bergeron [22] constructed a multiresolution display using wavelet approximations, where the data size is reduced by repeatedly merging neighboring points. Their approach was to construct hierarchical structures using the wavelet transform and view different levels of detail interactively upon the hierarchical structures. However, the wavelet transform requires the data to be ordered, making it useful only for data sets with a natural ordering, such as time-series data.

Another approach is to let the characteristics of the data set reveal themselves. For example, Wegman and Luo [20] suggest over-plotting translucent data points or lines so that sparse areas fade away while dense areas appear emphasized. The disadvantage of this method is that it relies on overlapping points or lines to identify clusters. Clusters without overlapping elements will not be visually emphasized.

Keim et al. [11] studied pixel-level visualization schemes which permit the display of a large number of records on a typical workstation screen based on recursive layout patterns. However, the number of displayable records is dependent on the size of the display area. This limitation restricts the scalability of their method. Moreover, since each pixel only represents one variable, it is difficult to convey the interactions among variables. We instead take a different approach of preserving the features of existing traditional display techniques as much as possible, and addressing the clutter problem by abstracting the

data itself into several levels of detail.

Wills [21] describes a visualization technique for hierarchical clusters. His approach expands upon the tree-map idea [15] by recursively subdividing the tree based on a dissimilarity measure. However, the main purpose is to display the clustering results, and in particular, the data partitions at a given dissimilarity value.

Many ideas in our work come from these and other research efforts. For example, we use the idea of the multiresolution display [22] in the IHD framework, while we uses hierarchical clustering [21] instead of the wavelet transform [22]. Inspired by Wegman and Luo [20], we use translucent-varying bands to indicate the extent of the clusters in our work.

Our starting point for the IHDs is, of course, the hierarchical parallel coordinates our team developed in 1999 [6,7]. The hierarchical parallel coordinates extend the parallel coordinates display technique by using a multi-resolution view of the data via hierarchical clustering. In this paper we generalized the principles underlying the hierarchical parallel coordinates by developing the general framework of IHDs from it. We then validated this framework by applying it successfully to the three other traditional flat display techniques in XmdvTool.

7 Conclusions and Future Work

In this paper, we presented a framework (Interactive Hierarchical Displays) for addressing the problem faced when attempting to visualize very large multivariate data sets. We described the full implementation of this framework and

its successful application to four traditional multivariate visualization techniques. Our modular design allowed us to integrate new display techniques without having to vary either the data structure or the interactive exploration tools. This, we feel, validates the generality of the IHD framework. We also conducted an empirical evaluation that verified the effectiveness of the interactive hierarchical displays.

This work is part of an ongoing research project at WPI focusing on multivariate visualization of large data sets. In the future, we want to further improve the interactive exploration tools to make them more user-friendly and flexible. We also want to improve the individual hierarchical display techniques to make them more powerful. For example, we hope to research different glyph placement strategies to get a suitable placement algorithm for hierarchically related glyphs. Finally, we are developing a suite of distortion techniques to allow users to selectively allocate more screen space to regions of interest, while deemphasizing the remaining data without loss of context. We feel this will be a powerful supplement to the IHD framework in facilitating exploratory analysis of large data sets.

Acknowledgements

The authors wish to thank Ying-Huey Fua, Daniel Stroe, Punit Doshi, and Yakov Kronrod for their assistance on the XmdvTool Project. They also wish to thank the National Science Foundation for supporting this research under grants IIS-9732897 and CISE-9729878.

References

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

[2] D.F. Andrews. Plots of high dimensional data. *Biometrics, Vol. 28, p. 125-36*, 1972.

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

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

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

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

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

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

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

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

[11] D.A. Keim, H.P. Kriegel, and M. Ankerst. Recursive pattern: a technique for visualizing very large amounts of data. *Proc. of Visualization '95, p. 279-86*, 1995.

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

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

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

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

[16] J.H. Siegel, E.J. Farrell, R.M. Goldwyn, and H.P. Friedman. The surgical implication of physiologic patterns in myocardial infarction shock. *Surgery Vol. 72, p. 126-41*, 1972.

[17] I. D. Stroe, E. A. Rundensteiner, and M. O. Ward. Scalable visual hierarchy exploration. In *Proc. Database and Expert Systems Applications, pp. 784-793*, 2000.

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

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

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

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

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

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

VITAE

Jing Yang is a Ph.D. student in the Computer Science Department at Worcester Polytechnic Institute (WPI). She received her B.S. degree from Tsinghua University in 1997 and worked in the National CAD Engineering Center of China prior to starting her graduate studies at WPI in 2000. Her research interests include Computer Graphics, Data Visualization, and Geometric Modeling.

Matthew Ward is a full professor in the Computer Science Department at WPI. He received his Ph.D. in Computer Science from the University of Connecticut in 1981. Prior to joining WPI in 1986 he was a Member of the Technical Staff in the Visual Communications Research Laboratory at Bell Laboratories and a Pattern Recognition Specialist at Skantek Corporation. His research interests include Data and Information Visualization, Exploratory Data Analysis, Computer Graphics, and Knowledge-Guided Image Analysis.

Elke Rundensteiner is an associate professor in the Computer Science Department at WPI. She received her Ph.D. in Computer Science from the University of California in 1992, and was an assistant professor in the Electrical Engineering and Computer Science Department at the University of Michigan prior to joining the WPI faculty in 1996. Her research interests include Database and Information Systems, Data Warehousing, Visual Database Interfaces, and Data and Knowledge Exploration.

CONTACT INFORMATION

Jing Yang

Department of Computer Science

Worcester Polytechnic Institute

Worcester, MA 01609 USA

yangjing@cs.wpi.edu

(508) 831-5952 (phone)

(508) 831-5776 (fax)

Matthew Ward (corresponding author)

Department of Computer Science

Worcester Polytechnic Institute

Worcester, MA 01609 USA

matt@cs.wpi.edu

(508) 831-5671 (phone)

(508) 831-5776 (fax)

Elke Rundensteiner

Department of Computer Science

Worcester Polytechnic Institute

Worcester, MA 01609 USA

rundenst@cs.wpi.edu

(508) 831-5815 (phone)

(508) 831-5776 (fax)