Clutter Reduction in Multi-Dimensional Data Visualization Using Dimension Reordering*

Wei Peng, Matthew O. Ward and Elke A. Rundensteiner
Computer Science Department, Worcester Polytechnic Institute, Worcester, MA 01609
{debbie,matt,rundenst}@cs.wpi.edu

ABSTRACT

Visual clutter denotes a disordered collection of graphical entities in information visualization. Clutter can obscure the structure present in the data. Even in a small dataset, clutter can make it hard for the viewer to find patterns, relationships and structure.

In this paper, we define visual clutter as any aspect of the visualization that interferes with the viewer's understanding of the data, and present the concept of clutter-based dimension reordering. Dimension order is an attribute that can significantly affect a visualization's expressiveness. By varying the dimension order in a display, it is possible to reduce clutter without reducing information content or modifying the data in any way.

Clutter reduction is a display-dependent task. In this paper, we follow a three-step procedure for four different visualization techniques. For each display technique, first, we determine what constitutes clutter in terms of display properties; then we design a metric to measure visual clutter in this display; finally we search for an order that minimizes the clutter in a display.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces I.5.3 [Pattern Recognition]: Clustering—Similarity Measures

Keywords: Multidimensional visualization, dimension order, visual clutter, visual structure.

1 INTRODUCTION

Visualization is the graphical presentation of information with the goal of helping the user gain a qualitative understanding of the information. A good visualization clearly reveals structure within the data and thus can help the viewer to better identify patterns and detect outliers. Clutter, on the other hand, is characterized by crowded and disordered visual entities that obscure the structure in visual displays. In other words, clutter is the opposite of structure; it corresponds to all the factors that interfere with the process of finding structures. Clutter is certainly undesirable since it hinders viewers' understanding of the content of the displays. However, when the dimensions or number of data items grow high, it is inevitable for displays to exhibit some clutter, no matter what visualization method is used.

To address this problem, many clutter reduction techniques have been proposed, such as multi-resolution approaches [23, 7, 24], dimensionality reduction approaches [9, 12, 22, 11], and distortion approaches [16, 14]. However, they either sacrifice the integrity of the data or fail to generate an unbiased representation of the data. In order to complement these approaches by reducing clutter in traditional visualization techniques while retaining the information in the display, we propose a clutter reduction technique using dimension reordering.

In many multivariate visualization techniques, such as parallel coordinates [8, 21], glyphs [1, 15], scatterplot matrices [3] and pixel-oriented methods [10], dimensions are positioned in some one- or two-dimensional arrangement on the screen [25]. Given the 2-D nature of this medium, some ordering or organization of the dimensions must be assumed. This organization can have a major impact on the expressiveness of the visualization. Different orderings of dimensions can reveal different aspects of the data and affect the perceived clutter and structure in the display. Thus completely different conclusions may be drawn based on each display. Unfortunately, in many existing visualization systems that encompass these techniques, dimensions are usually ordered without much care. In fact, dimensions are often displayed by the default order in the original dataset. Manual dimension ordering is available in some systems. For example, Polaris [18] allows users to manually select and order the dimensions to be mapped to the display. Similarly, in XmdvTool [25], users can manually change the order of dimensions from a reconfigurable list of dimensions. However, the exhaustive search for the best ordering is tedious even for a modest number of dimensions. Therefore, automatic clutter-based dimension ordering techniques that would remedy this shortcoming of current tools are needed.

Clutter reduction is a visualization-dependent task because visualization techniques vary largely from one to another. The basic goal of this paper is to present clutter measuring and reduction approaches for several of the most popular visualization techniques, namely parallel coordinates [8, 21], scatterplot matrices [3], star glyphs [17], and dimensional stacking [13]. Although we only chose these visualization techniques to experiment with, there are many more traditional visualization techniques that could benefit from this concept.

In order to automate the dimension reordering process for a display, we are concerned with three issues: (1) determining the way clutter manifests itself in the display, (2) designing a metric to measure visual clutter, and (3) arranging the dimensions for the purpose of clutter reduction. The solutions we provide are specifically tuned to each individual visualization technique. In some techniques, we reduce the level of noise in the display; in other cases we increase the number of clusters. For each technique we will follow a similar procedure. First we determine the visual characteristics that we would label as clutter. Next, we carefully define a metric for measuring clutter. Then we find the dimension order that minimizes the clutter in a display.

The remainder of this paper is organized as follows. Section 2 provides a review of related work. Sections 3, 4, 5, and 6 discuss the clutter definitions and measures for four different visualization techniques respectively. In Section 7, algorithms for reordering are presented. Conclusions and future work are presented in Section 8.

2 RELATED WORK

Many approaches have been proposed to overcome the clutter problem. Distortion [16, 14] is a widely used technique for clutter reduc-

*This work was supported under NSF grant IIS-0119276.

tion. In visualizations supporting distortion-oriented techniques, the interesting portion of the data is given more display space. The problem with this technique is that the uninteresting subset of the data is squeezed into a small area, making it difficult for the viewer to fully understand it. Multi-resolution approaches [23, 7, 24] are used to group the data into hierarchies and display them at a desired level of detail. These approaches do not retain all the information in the data, since many details will be filtered out at low resolutions.

High dimensionality is another source of clutter. Many approaches exist for dimension reduction. Principal Component Analysis [9], Multi-dimensional Scaling [12, 22], and Self Organizing Maps [11] are popular dimensionality reduction techniques used in data and information visualization. Yang et al. [27] proposed a visual hierarchical dimension reduction technique that creates meaningful lower dimensional spaces with representative dimensions from the original data space instead of generating new dimensions. These techniques generate a lower dimensional subspace to reduce clutter but some information in the original data space is also lost.

In information visualization, many visual factors can be ordered to enhance the displays. Friendly et al. [6] designed a general framework for ordering information, including arrangement of variables, according to the desired effects or trends. Dimension ordering has also been studied in [2, 26]. Ankerst et al. [2] proposed a method to arrange dimensions according to their similarities so that similar ones are adjacent to each other. They used Euclidean distance as the similarity measure, proved that the arrangement problem is NP-complete, and applied heuristic algorithms to search for the optimal order. Yang et al. [26] imposed a hierarchical structure over the dimensions themselves, grouping a large number of dimensions into a hierarchy so that the complexity of the ordering problem is reduced. User interactions are then supported to make it practical for users to actively decide on dimension reduction and ordering in the visualization process. However, in those approaches, dimensions are reordered according to only one particular measure, the similarity between dimensions. In many visualization techniques, the overall clutter in the display is not always related to similarity between dimensions. Ordering dimensions according to the best correlation does not guarantee the least clutter. But their idea of using dimension ordering inspired our work of ordering dimensions to improve visualization quality.

3 Parallel Coordinates

Parallel coordinates is a popular multivariate visualization technique [8, 21]. In this method, each dimension corresponds to an axis, and the N axes are organized as uniformly spaced vertical or horizontal lines. A data element in an N-dimensional space manifests itself as a connected set of points, one on each axis. Thus a polyline is generated for representing one data point.

3.1 Clutter Analysis of Parallel Coordinates

In the parallel coordinates display, as the axes order is changed, the polylines representing data points take on very distinct shapes. In Figures 1 and 2, the two displays depict the same dataset with different dimension orders. As can be seen in the figure, a parallel coordinates display makes inter-dimensional relationships between neighboring dimensions easy to see, but does not disclose relationships between non-adjacent dimensions. In a full display of parallel coordinates without sampling, filtering or multi-resolution processing, if polylines between two dimensions can be naturally grouped into a set of clusters, the user will likely find it easier to comprehend the relationship between them. Instead, if there are many lines that don't belong to any cluster, the space between the two dimensions can be very cluttered. These polylines don't help the viewer to find patterns and discover relationships. Those data points that don't belong to any cluster are called outliers. It is true that one of the advantages of parallel coordinates visualization is to help find outliers, but in our case, a lot of outliers between a pair of dimensions indicates that there is little relationship between the two of them. Since our goal is to disclose more relationships and patterns between dimensions, we want to minimize the impact from outliers, in other words, we carefully order the dimensions to avoid them.

3.2 Clutter Measure in Parallel Coordinates

3.2.1 Defining and Computing Clutter

Due to the fact that outliers often obscure structure and thus confuse the user, clutter in parallel coordinates can be defined as the proportion of outliers against the total number of data points. To reduce clutter in this technique, our task is to rearrange the dimensions to minimize the outliers between neighboring dimensions. To calculate the score for a given dimension order, we first count the total number of outliers between neighboring dimensions, $S_{outlier}$. If there are n dimensions, the number of neighboring pairs for a given order is $n-1$. The average outlier number between dimensions is defined to be $S_{avg} = S_{outlier}/(n-1)$. Let S_{total} denote the total number of data points. The clutter $\mathscr{C}$, defined as the proportion of outliers, can then be calculated as follows:

$$\mathscr{C} = S_{avg}/S_{total} = \frac{\frac{S_{outlier}}{n-1}}{S_{total}} \tag{1}$$

Since $n-1$ and S_{total} are both fixed for a given dataset, dimension orders that reduce the total number of outliers also reduce clutter in the display according to our notion of clutter.

Now we are faced with the problem of how to decide if a data item is within a cluster or is an outlier. Since we have restricted the notion of clutter to the number of outliers within neighboring pairs of dimensions, we can use the normalized Euclidean distances between data points to measure their closeness. If a data point does not have any neighbor whose distance to it is less than threshold t, we treat it as an outlier. In this way, we are able to find all the data points that don't have any neighbors within the distance t in the specified two-dimensional space. If the number of data points is m, this is done in $O(m^2)$ time. We do this for every pair of the n dimensions and store the outlier numbers in a outlier matrix M. The total time for building this matrix is $O(m^2n^2)$. Given a dimension order, we can then decide the clutter in the display by adding up outlier numbers between neighboring dimensions.

Instead of letting the user specify the threshold, we could have decided it based on the dataset, or develop algorithms that don't involve thresholds. However, since we want to give the user more flexibility and interaction when ordering the dimensions, we believe that allowing the user to decide the thresholds of cluster width is preferable. Thus the threshold here and those in the following chapters all can be user-defined, though with a fixed default value.

3.2.2 The Optimal Dimension Order

Optimal dimension ordering would be to select the one dimension order that minimizes visual clutter. In a given dimension order, adding up outlier numbers between neighboring dimensions takes $O(n)$ time. Since the optimal dimension ordering algorithm is an exhaustive search algorithm with $O(n!)$ time, the search time involved is $O(n*n!)$.

3.3 Example

Figures 1 and 2 both represent the Cars dataset. In Figure 1 the data is displayed with the default dimension ordering. Figure 2 displays

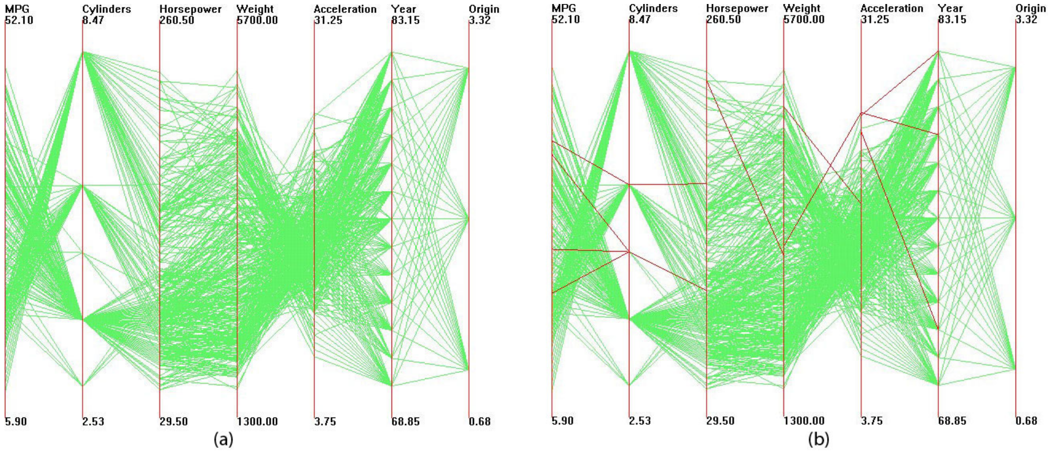


Figure 1: Parallel coordinates visualization of Cars dataset. Outliers are highlighted with red in (b).

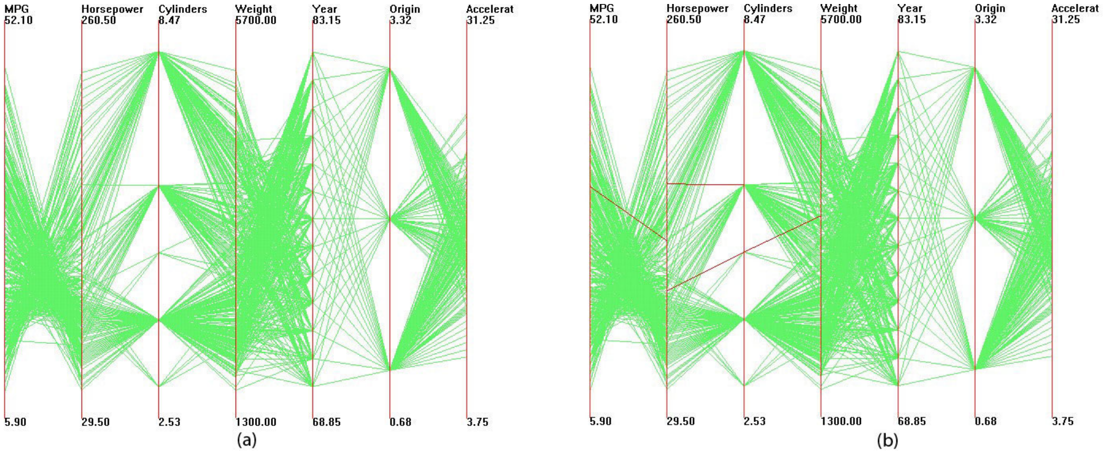


Figure 2: Parallel coordinates visualization of Cars dataset after clutter-based dimension reordering. Outliers are highlighted with red in (b).

the data after being processed with clutter-based ordering. In the rightmost image in each figure, polylines highlighted in red are outliers according to our clutter metric. With a glimpse we can identify more outliers in the original visualization than the improved one. It is also clear that, in the new display, neighboring dimensions are more tightly related. In addition, the data points are better separated and thus it is easier for the viewer to find patterns.

4 Scatterplot Matrices

Scatterplot matrices are one of the oldest and most commonly used methods to project high dimensional data to 2-dimensions [1]. In this method, $N*(N-1)/2$ pairwise parallel projections are generated, each giving the viewer a general impression regarding relationships within the data between pairs of dimensions. The projections are arranged in a grid structure to help the user remember the dimensions associated with each projection.

4.1 Clutter Analysis in Scatterplot Matrices

In clutter reduction for scatterplot matrices, we focus on finding structure in plots rather than outliers, because the overall shape and tendency of data points in a plot can reveal a lot of information. Some work has been done in finding structures in scatterplot visualizations. PRIM-9 [19] is a system that makes use of scatterplots. In PRIM-9 [19] data is projected onto a two-dimensional subspace defined by any pair of dimensions. Thus the user can navigate all the projections and search for the most interesting ones. Automatic projection pursuit techniques [5] utilize algorithms to detect structure in projections based on the density of clusters and separation of data points in the projection space to aid in finding the most interesting plots.

With a matrix of scatterplots, users are not only able to find plots with structure, but also can view and compare the relationships between these plots. Since all orthogonal projections are displayed on the screen, changing the dimension order does not result in different projections, but rather a different placement of the pairwise plots. In practice, it will be beneficial for the user to have projections that disclose a related structure to be placed next or close to each other in order to reveal important dimension relationships in the data. To make this possible, we have defined a clutter measure for scatterplot matrices. The main idea is to find the structure in all 2-dimensional projections and use it to determine the position of dimensions so that plots displaying a similar structure are positioned near each other.

Figure 3 gives two views of a scatterplot matrix visualization. In this type of visualization, we can separate the dimensions into two categories: high-cardinality dimensions and low-cardinality dimensions. In high-cardinality dimensions, data values are often contin-

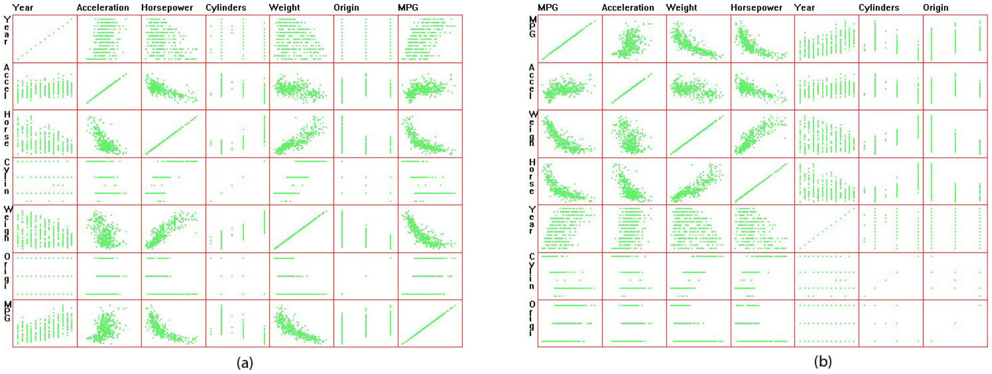


Figure 3: Scatterplot matrices visualization of Cars dataset. In (a) dimensions are randomly positioned. After clutter reduction (b) is generated. The first four dimensions are ordered with the high-cardinality dimension reordering approach, and the other three dimensions are ordered with low-cardinality approach.

uous, such as height or weight, and can take on any real number within the range. In low-cardinality dimensions, data values are often discrete, such as gender, type, and year. These data points often take a small number of possible values. It is often perceived that plots involving only high-cardinality dimensions will place dots in a scattered manner while plots involving low-cardinality dimensions will place dots in straight lines because a lot of data points share the same value on this dimension. In this paper, we determine if a dimension is high or low-cardinality depending on the number of data points and their possible values. Let m_i denote the number of possible data values on the ith dimension, and m denote the total number of data points. If $m_i \geq \sqrt{m}$, dimension i is considered to be of high-cardinality, otherwise it is low-cardinality.

We will treat high-cardinality and low-cardinality dimensions separately because they generate different plot shapes. The clutter definition and clutter computation algorithms for these two classes of dimensions will differ from each other.

4.2 High-Cardinality Clutter Measure in Scatterplot Matrices

4.2.1 Defining and Computing Clutter

The correlation between two variables reflects the degree to which the variables are associated. The most common measure of correlation is the Pearson Correlation Coefficient, which can be calculated as:

$$r = \frac{\sum_i (x_i - x_m)(y_i - y_m)}{\sqrt{\sum_i (x_i - x_m)^2}\sqrt{\sum_i (y_i - y_m)^2}} \tag{2}$$

where x_i and y_i are the values of the ith data point on the two dimensions, and x_m and y_m represent the mean value of the two dimensions.

Since plots similarly correlated will likely display a similar pattern and tendency, we can calculate the correlations for all the two-dimensional plots (in fact half of them because the matrix is symmetric along the diagonal), and reorder the dimensions so that plots whose correlation differences are within threshold t are displayed as close to each other as possible. To achieve this goal, we define the sum of the distances between similar plots to be the clutter measure. In our implementation, we define the plot side length to be 1 and calculate the distance between plots X and Y using $\sqrt{(Row_X - Row_Y)^2 + (Column_X - Column_Y)^2}$. For example, in Figure 4, the distance between similar plots A and B will be $\sqrt{(1-0)^2+(1-0)^2} = \sqrt{2}$. Larger distance sum means similar plots are more scattered in the display, thus the view is more cluttered.

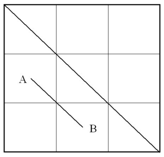


Figure 4: Illustration of distance calculation in scatterplot matrices.

In the high-cardinality dimension space, our approach to calculate total clutter for a certain dimension ordering is as follows. Let p_i be the ith plot we visit. Let threshold t be the maximum correlation difference between plots that can be called ”similar”. Note that we are only concerned with the lower-left half of the plots, because the plots are symmetric along the diagonal. The plots along the diagonal will not be considered because they only disclose the correlations of dimensions with themselves. This is always 1.

In a fixed matrix configuration, we do the following to compute the clutter of the display. First, a correlation matrix $M(n,n)$ is generated for all n high-cardinality dimensions. $M[i][j]$ represents the Pearson correlation coefficient for the plot on the ith row and jth column. If data number is m, the complexity of building up this matrix is $O(m*n^2)$. Then, for any plot p_i, we find all the plots that have a similar correlation with it, i.e, the differences between their Pearson correlation coefficients with p_i's are within threshold t. This process will take $O(n^3)$. We store this information so we only have to do it once.

4.2.2 The Optimal Dimension Order

We get a total distance for any scatterplot matrix display. With this measure, comparisons between different displays of the same data can be made. Unlike the one-dimensional parallel coordinates display, we have to calculate distances for every pair of plots. If a pair of plots has similar correlation, their distance is added to the total clutter measure of the display. This is an $O(n^2)$ process. An optimal dimension order can be achieved by an exhaustive search with complexity $O(n!)$. Therefore the total computing time will be $O(n^2*n!)$.

4.3 Low-Cardinality Clutter Measure in Scatterplot Matrices

4.3.1 Defining and Computing Clutter

In low-cardinality dimensions, we also want to place similar plots together. But we use a different clutter measure from high-cardinality dimensions.

For plots with low-cardinality dimensions, the higher the cardinality, the more crowded the plot seems to be. Therefore, instead of navigating all dimension orders and searching for the best one, we will order these dimensions according to their cardinalities. Dimensions with higher cardinality are positioned before lower-cardinality dimensions. In this way, plots with similar density are placed near each other. This satisfies our purpose for clutter reduction. The dot density of plots will appear to decrease gradually, resulting in less clutter, or more perceived order, in the view.

4.3.2 The Optimal Dimension Order

With low-cardinality dimensions, the dimension reordering can be envisioned as a sorting problem. With a quick sort algorithm, it can be achieved within $O(n*\log n)$ time.

4.4 Example

From Figure 3 we notice that plots generated by two high-cardinality dimensions are very different in pattern with plots involving one or two low-cardinality dimensions. We believe that separating the high and low-cardinality dimensions from each other is useful in identifying similar low-cardinality dimensions and finding similar plots in the high-cardinality dimension subspace.

5 Star Glyphs

5.1 Clutter Analysis in Star Glyphs

A glyph is a representation of a data element that maps data values to various geometric and color attributes of graphical primitives or symbols [15]. XmdvTool [25] uses star glyphs [17] as one of its four visualization approaches. In this technique, each data element occupies one portion of the display window. Data values control the length of rays emanating from a central point. The rays are joined by a polyline drawn around the outside of the rays to form a closed polygon.

In star glyph visualization, each glyph represents a different data point. With dimensions ordered differently, the glyph's shape varies. Since glyphs are stand-alone graphical entities, we consider reducing clutter here as to make those single data points overall seem more structured. Alternatively we could have focused on glyph placement as a means of reducing clutter. Gestalt Laws are robust rules of pattern perception [20]. They state that similarity and symmetry are two factors that help viewers see patterns in the visual display. We call a glyph well structured if its rays are arranged so that they have similar length to their neighbors and are well balanced along some axis. In our approach, we define monotonicity and symmetry as our measures of structure for glyphs. Therefore user can find monotonic structure, symmetric structure, or a combination of the two in the data.

Let's take monotonicity+symmetry for example. In a perfectly structured glyph:

- Neighboring rays have similar lengths.
- The lengths of rays are ordered in a monotonically increasing or decreasing manner on both sides of an axis.
- Rays of similar lengths are positioned symmetrically along either a horizontal or vertical axis.

The perfectly structured star glyph is thus a teardrop shape. With such shapes in glyphs, the user will find it easier to identify relative value differences between dimensions, and can better discern rays and the bounding polylines. For instance, the data points shown in Figure 5 present very different shapes with different dimension order. The original order in Fig.5-(a) makes them look irregular and display a concave shape, while the dimension order in Fig.5-(b) makes them more symmetric and easy to interpret.

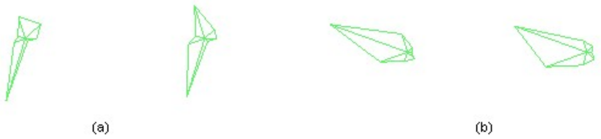


Figure 5: The two glyphs in (a) represent the same data points as (b), with a different dimension order.

5.2 Clutter Measure in Star Glyphs

5.2.1 Defining and Computing Clutter

To reduce the clutter for the whole display, we seek to reorder the dimensions to minimize the total occurrence of unstructured rays in glyphs. Therefore, we define clutter as the total number of non-monotonic and non-symmetric occurrences. We believe that with more rays in data points displaying a monotonic and symmetric shape, the structure in the visualization will be easier to perceive.

In order to calculate clutter in one display, we test every glyph for its monotonicity and symmetry. Suppose the user chooses both monotonicity and symmetry as the structure measure, and specifies the first half of the dimensions being monotonically increasing and the second half of the dimensions being monotonically decreasing. The user can then choose a threshold $t1$ for checking monotinicity, and a threshold $t2$ for checking symmetry. $t1$ and $t2$ are measures for normalized numbers and thus can take any number from 0 to 1. Suppose a point has normalized values on two neighboring dimensions ($dimension_{n-1}$ and $dimension_0$ are not considered neighbors), p_i and p_{i+1}. If the two values don't violate the user's specification for monotonicity, nothing happens. However, if the two values violate the user's specification for monotonicity, we will check their difference and decide if they clutter the view or not. For instance, if p_{i+1} is less than p_i while $dimension_i$ and $dimension_{i+1}$ are among the first half of the dimensions, it is a violation of the monotonicity rule. We will see if $p_i - p_{i+1}$ is less than threshold $t1$ or not. If so, we consider this non-monotonicity occurrence as tolerable, and still, nothing happens. If not, we will add this occurrence to our measure count of unstructuredness. Similarly, for two dimensions that are symmetrically positioned along the horizontal axis, if their difference is within threshold $t2$, they are considered symmetric to each other. Otherwise another increment is added to the total occurrence of unstructuredness.

5.2.2 The Optimal Dimension Order

The calculation for a single glyph involves going through $n-1$ pairs of neighboring dimensions to check for monotonicity and $n/2$ pairs of dimensions symmetric along the axis. Therefore, for a dataset with m data points, the calculation takes $O(n*m)$. With the exhaustive search for best ordering, the computational complexity for dimensional reordering in star glyphs is $O(n*m*n!)$.

5.3 Example

For each ordering we can count the unstructuredness occurrences to find the order that minimizes this measure. Figure 6 displays the Coal Disaster dataset before and after clutter reduction. In Fig.6-(a), many glyphs are displayed in a concave manner, and it's hard

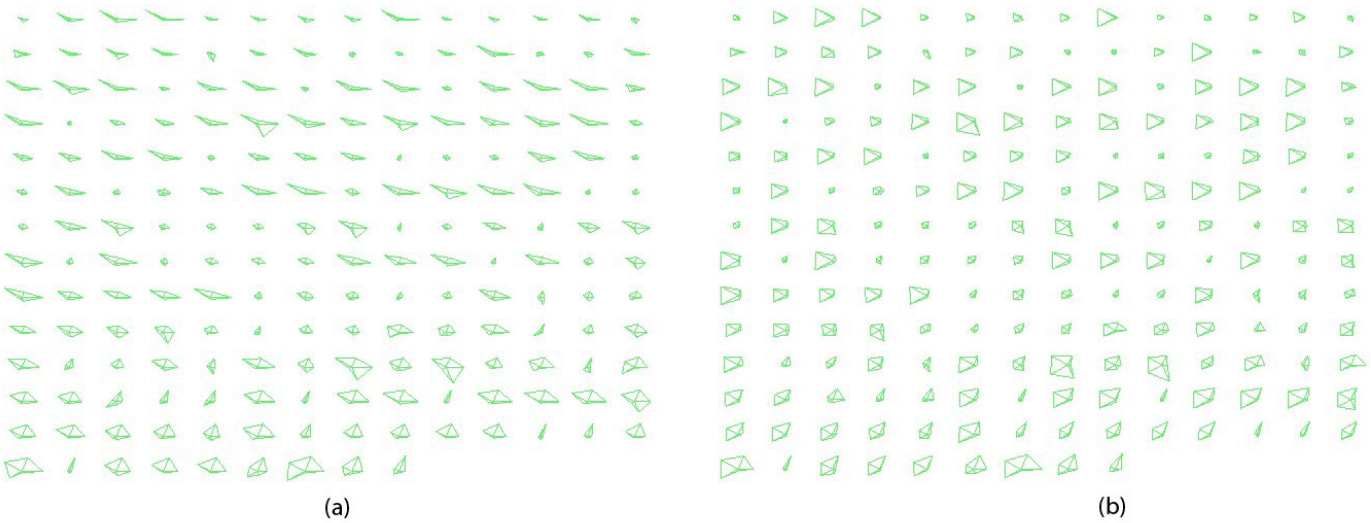


Figure 6: Star glyph visualizations of Coal Disaster dataset. (a) represents the data with original dimension order, and (b) shows the data after clutter is reduced. The shapes on (b) should mostly appear simpler.

to tell the dimensions from bounding polylines. This situation is improved in Fig.6-(b) with clutter-based dimension reordering.

6 Dimensional Stacking

6.1 Clutter Analysis in Dimensional Stacking

The dimensional stacking technique is a recursive mapping method developed by LeBlanc et al. [13]. Each dimension of the dataset is first discretized into a number of bins. Then two dimensions are defined as the horizontal and vertical axis, creating a grid on the display. Within each box of this grid this process is applied again with the next two dimensions. This process continues until all dimensions are assigned. Each data point maps to a single bin based on its values in each dimension.

Figure 7-(a) illustrates the Iris dataset with original dimension order, i.e., dimensions in the order: sepal length, sepal width, petal length and petal width, represented in this display as "outer horizontal", "outer vertical", "inner horizontal" and "inner vertical" respectively. Each of the four dimensions is divided into five bins (ranges of values).

In this technique, the dimension order determines the orientation of axes and the number of cells within a grid. The inner-most dimensions are named the fastest dimensions because along these dimensions two small bins immediately next to each other represent two different ranges of the dimensions. On the contrary, the outermost dimensions have the slowest value changing speed, meaning many neighboring bins on these dimensions are within the same value range. Therefore, in dimensional stacking, the order of dimensions has a huge impact on the visual display.

For dimensional stacking, the bins within which data points fall are shown as filled squares. These bins naturally form groups in the display. We hypothesize that a user will consider a dimensional stacking visualization as highly structured if it displays these squares mostly in groups. Compared to a display with mainly randomly scattered filled bins, those that contain a small number of groups appear to have more structure and thus can be better interpreted. The data points within a group share similar attributes in many aspects. Thus this view will help the user to search for groupings in the dataset as well as to detect subtle variances within each group of data points. The other data points that are considered outliers may also be readily perceived if most data falls within a small number of groups.

6.2 Clutter Measure in Dimensional Stacking

6.2.1 Defining and Computing Clutter

We define the clutter measure as the proportion of occupied bins aggregated with each other versus small isolated "islands", namely the filled bins without any neighbors around them. A measure of clutter might then be $\frac{number\ of\ isolated\ filled\ bins}{number\ of\ total\ occupied\ bins}$. The dimension order that minimizes this number will then be considered the best order. The user can define how large a cluster should be for its member bins to be considered "clustered" or "isolated". Besides that, we need to also define which bins are considered neighbors. The choices are 4-adjacent bins and 8-adjacent bins. 4-adjacency and 8-adjacency are terms from image science, which help to define neighbors of pixels. Since we are dealing with bins in grids that are quite similar to pixels in images, we employ the concept here. Two bins are 4-adjacent if they lie next to each other horizontally or vertically, while they are 8-adjacent if they lie next to one another horizontally, vertically, or diagonally. With 4-adjacency being used, the adjacent bins would share the same data range on all but one dimension, while the 8-adjacent bins may fall into different data ranges on at most two dimensions.

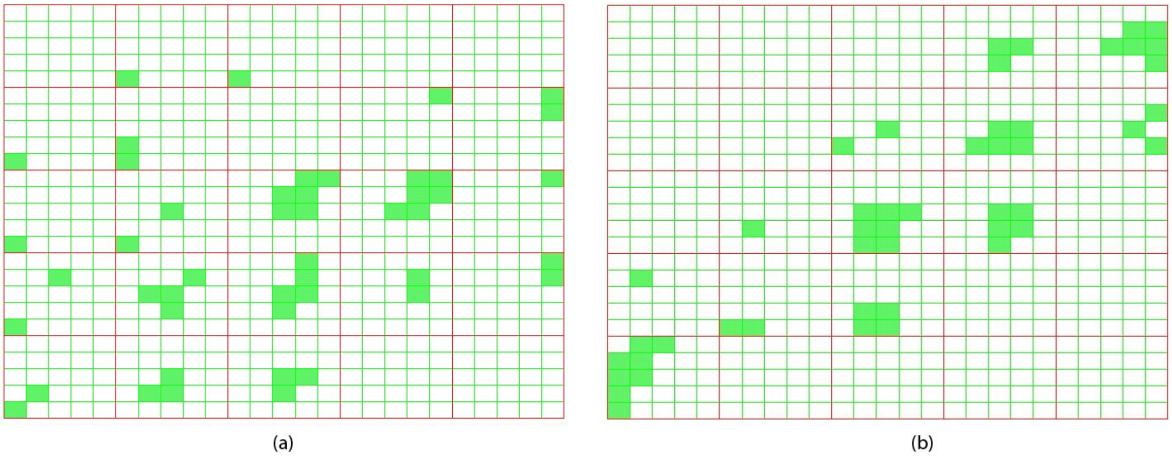
Given a dimension order, our approach will search for all filled bins that are connected to neighbors and calculate clutter according to the above clutter measure. The dimension order that minimizes this number is considered the best ordering.

6.2.2 The Optimal Dimension Order

This algorithm is similar to that used with high-cardinality dimensions in scatterplot matrices. However we are comparing the position of bins instead of plots. The computational complexity will be $O(m^2)$ for one dimension order. The optimal search would thus take $O(m^2 * n!)$.

6.3 Example

An example of clutter reduction in dimensional stacking is given in Figure 7. We use 8-adjacent neighbors in our calculation. Fig.7-(a),


(a)

(b)

Figure 7: Dimensional stacking visualization for Iris dataset. (a) represents the data with original dataset, and (b) shows the data with clutter reduced.

Table 1: Table of computation times using optimal ordering algorithm

Visualization	Algorithm Complexity	Dataset	Data Number	Dimensionality	Time
Parallel Coordinates	$O(n*n!)$	AAUP-Part	1161	9	3 sec.
		Cereal-Part	77	10	23 sec.
		Voy-Part	744	11	4:02 min.
Scatterplot Matrices	$O(n^2*n!)$	Voy-Part	744	11(6 high-card dimensions)	5 sec.
		AAUP-Part	1161	9	3:13 min.
Star Glyphs	$O(m*n*n!)$	Cars	392	7	18 sec.
Dimensional Stacking	$O(m^2*n!)$	Coal Disaster	191	5	10 sec.
		Detroit	13	7	5 sec.

denoting the original data order, is composed of many “islands”, namely the filled bins without any occupied neighbors. In Fig.7-(b), the display has the optimal ordering: petal length, petal width, sepal length, sepal width. We can discover that there are fewer “islands”, and the filled bins are more concentrated. This helps us to see groups better than in the original order. In addition, the bins are distributed closely along the diagonal, which implies a tight correlation between the first two dimensions: petal length and petal width.

7 Analysis of Reordering Algorithms

As stated previously, the clutter measuring algorithms for the four visualization techniques take different amount of time to complete. Let m denote the data size, and n denote the dimensionality. The computational complexity of measuring clutter in the four techniques is presented in Table 1.

Exhaustive search would guarantee the best dimension order that minimizes the total clutter in the display. However, in [2], Ankerst et al. proved that an optimal search for best dimension order is an NP-complete problem, equivalent to the Brute-Force solution to Traveling Salesman Problem. Therefore, we can do the optimal search with only low dimensionality datasets. To get a quantitative understanding of this issue, we performed a few experiments for different visualizations, and the results obtained are presented in Table 1. We realized that even in a low dimensional data space - around 10 dimensions - the computational overhead could be significant. If the dimension number exceeds that, we need to resort to heuristic approaches. For example, random swapping, nearest-neighbor and greedy algorithms have been implemented and tested.

The random swapping algorithm starts with an initial configuration and randomly chooses two dimensions to switch their positions. If the new arrangement results in less clutter, then this arrangement is kept and the old one is rejected; otherwise the old arrangement is left intact and another pair of dimensions are swapped. Keep doing this until no better result is generated for a certain number of swaps. This algorithm can be applied to all the visualization techniques.

The nearest-neighbor algorithm starts with an initial dimension, finds the nearest neighbor of it, and adds the new dimension into the tour. Then, it sets the new dimension to be the current dimension for searching neighbors. Continue until all the dimensions have been added into the tour. The greedy algorithm [4] keeps adding the nearest possible pairs of dimensions, until all the dimensions are in the tour.

The nearest-neighbor and greedy algorithms are good for parallel coordinates and scatterplot matrices displays. In those displays, there is some overall relationship between dimensions that can be calculated, such as the number of outliers between dimensions and correlation between dimensions. However, in the star glyph and dimensional stacking visualizations, we calculate the clutter of a display under certain dimension arrangements, instead of defining relationship between each two dimensions. Thus these algorithms are not very amenable to the latter two techniques.

With these heuristic algorithms, we perform dimension reordering for datasets with much higher dimensions with relatively good results. Experimental results are presented in Table 2.

Table 2: Table of computation times using heuristic algorithms

Visualization	Dataset	Data Number	Dimensionality	Algorithm	Time
Parallel Coordinates	Census-Income	200	42	Nearest-Neighbor Algorithm	2 sec.
				Greedy Algorithm	3 sec.
				Random Swapping	2 sec.
	AAUP	1161	14	Nearest-Neighbor Algorithm	7 sec.
				Greedy Algorithm	9 sec.
				Random Swapping	6 sec.
Scatterplot Matrices	Census-Income	200	42	Nearest-Neighbor Algorithm	2 sec.
				Greedy Algorithm	3 sec.
				Random Swapping	2 sec.
	AAUP	1161	14	Nearest-Neighbor Algorithm	8 sec.
				Greedy Algorithm	8 sec.
				Random Swapping	7 sec.
Star Glyphs	Census-Income	200	42	Random Swapping	2 sec.
	AAUP	1161	14	Random Swapping	7 sec.
Dimensional Stacking	Those datasets are too big for dimensional stacking visualization.				

8 Conclusion and Future Work

In this paper, we have proposed the concept of visual clutter measurement and reduction using dimension reordering in multi-dimensional visualization. We studied four rather distinct visualization techniques for clutter reduction. For each of them, we analyzed its characteristics and then defined an appropriate measure of visual clutter. In order to obtain the least clutter, we searched for a dimension order that minimizes the clutter in the display.

This represents a first step into the field of automated clutter reduction in multi-dimensional visualization. There are many visualization techniques that we haven't experimented with yet, and certainly our clutter measures are not the only ones possible. Our hope is to give users the ability to generate views of their data that will enable them to discover structure that they will otherwise not find in a view with the original or a random dimension order.

Future work will include the combination of clutter reduction approaches with dimension reduction or hierarchical data visualization to gauge the effectiveness of these techniques in high-dimensional or high data volume datasets.

References

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

[2] M. Ankerst, S. Berchtold, and D.A. Keim. Similarity clustering of dimensions for an enhanced visualization of multidimensional data. *Proc. IEEE Symp. on Information Visualization*, pages 52–60, 1998.

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

[4] Thomas H. Cormen, E. Leiserson, Charles, and Ronald L. Rivest. *Introduction to Algorithms*. MIT Press, 1990. COR t 01:1 1.Ex.

[5] S.L. Crawford and T.C. Fall. Projection pursuit techniques for visualizing high-dimensional data sets. *Visualization in Scientific Computing, (G.M. Nielson and B. Shriver, eds.)*, pages 94–108, 1990.

[6] M. Friendly and E. Kwan. Effect ordering for data displays. *Computational Statistics & Data Analysis*, 43(4):509–539, 2003.

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

[8] A. Inselberg. The plane with parallel coordinates. *Special Issue on Computational Geometry, The Visual Computer*, 1:69–97, 1985.

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

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

[11] T. Kohonen. The self-organizing map. *Proc. IEEE*, pages 1464–1480, 1978.

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

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

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

[15] R.J. Littlefield. Using the glyph concept to create user-definable display formats. *Proc. NCGA*, pages 697–706, 1983.

[16] M. Sheelagh, T. Carpendale, D.J. Cowperthwaite, and F.D. Fracchia. Distortion viewing techniques for 3-dimensional data. *Proc. IEEE Symposium on Information Visualization*, pages 46–53, 1996.

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

[18] 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.

[19] J.W. Tukey, M.A. Fisherkeller, and J.H. Friedman. Prim-9: An interactive multidimensional data display and analysis system. *Dynamic Graphics for Statistics, (W. S. Cleveland and M. E. McGill, eds.)*, pages 111–120, 1988.

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

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

[22] S.L. Weinberg. An introduction to multidimensional scaling. *Measurement and evaluation in counseling and development*, 24:12–36, 1991.

[23] P.C. Wong and R.D. Bergeron. Multiresolution multidimensional wavelet brushing. *Proc. IEEE Visualization*, pages 141–148, 1996.

[24] Allison Woodruff, James Landay, and Michael Stonebraker. Constant information density in zoomable interfaces. In *Proc. of the working conference on Advanced visual interfaces*, pages 57–65, 1998.

[25] Xmdvtool home page. http://davis.wpi.edu/˜xmdv/. http://davis.wpi.edu/˜xmdv.

[26] J. Yang, W. Peng, M.O. Ward, and E.A. Rundensteiner. Interactive hierarchical dimension ordering, spacing and filtering for exploration of high dimensional datasets. *Proc. IEEE Symposium on Information Visualization*, pages 105–112, 2003.

[27] J. Yang, M.O. Ward, E.A. Rundensteiner, and S. Huang. Visual hierarchical dimension reduction for exploration of high dimensional datasets. *Eurographics/IEEE TCVG Symposium on Visualization*, pages 19–28, 2003.