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

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 and reveal relationships. In this paper, we present the concept of clutter-based dimension reordering. Our hope is to reduce clutter without reducing information content or disturb data in any way. Dimension order is a variable that can significantly affect a visualization's expressiveness. By varying the dimension order in visualizations, our goal is to find the views with the least amount of visual clutter. Clutter reduction is a display-dependent task. We define different measures of what constitutes clutter in terms of display properties for four different visualization techniques. We then apply dimension ordering algorithms to search for a order that minimizes the clutter in a display.

CR Categories: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Graphical user interfaces H.2.8 [Database Management]: Database Applications—Data mining I.5.3 [Pattern Recognition]: Clustering—Similarity Measures

Keywords: multidimensional visualization, dimension ordering, visual clutter, visual structure

1 INTRODUCTION

Visualization is the graphical presentation of information, with the goal of facilitating the user to 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. Clutter is certainly undesirable since it hinders viewers' understanding of the displays content. However, when the dimensions or number of data items grow high, it is inevitable for users to encounter clutter, no matter what visual method is used.

Many clutter reduction techniques deal with data of high volume or high dimensionality, such as hierarchical clustering, sampling, and filtering. But they may result in some information loss. Distortion is another category of methods for clutter reduction. But distorted views do not give an unbiased representation of the data content because spatial relationships are modified. In order to complement these approaches, helping the user to reduce clutter in some 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 [6], glyphs [14], scatterplot matrices [1] and pixel-oriented methods [9], dimensions are positioned in some one- or two-dimensional arrangement on the screen [24]. Given the 2-D nature of this medium, the arrangement must choose some order of dimension. This arrangement can have a major impact on the expressiveness of the visualization. Different orders of dimensions can reveal different aspects of the data and affect the perceived clutter and structure in the display. Thus completely different conclusions can be drawn based on the available display. Unfortunately, in many existing visualization systems that encompass these techniques, dimensions are usually ordered without much care. In fact, dimensions are often determined 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 [24], users can manually change the order of dimensions from a reconfigurable list of dimensions. However, the exhaustive search for best ordering is tedious even for a modest number of dimensions. At the same time it lacks a quantitative measurement of quality. Therefore, automatic clutter-based dimension ordering techniques would remedy this shortcoming of current tools.

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 visualization techniques, namely parallel coordinates [6], scatterplot matrices [1], star glyphs [17], and dimensional stacking [12].

In order to automate the dimension reordering process for a display, we are concerned with two issues: (1) designing a metric to measure visual clutter, and (2) developing an algorithm to reorder the dimensions for the purpose of clutter reduction. The solutions we provide must be specifically tuned to each individual visualization technique. In some techniques, we want to reduce the level of noise that tends to obscure the structure in the display; in other cases we want to increase the number of clusters. In some cases we even want to do both. For each technique, we will study both issues. First, we will carefully define a metric for measuring clutter. Second we will choose one algorithm from the possible solution candidates to arrange the dimensions. Third, we will compare the results with the original display.

Our technique targets on small to middle-size dataset in terms of dimensionality. Although we only chose four visualization techniques to experiment with, there are many more traditional visualization techniques can benifit from this concept.

The remainder of this paper is organized as follows. Section 2 will provide 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.

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

2 RELATED WORK

To overcome the clutter problem, many approaches have been proposed. Distortion [16, 13] is a widely used technique used for clutter reduction. 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, 22, 5] are used to group the data into hierarchical clusters 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 [8], Multi-dimensional Scaling [11, 21], and Self Organizing Maps [10] are popular dimensionality reduction techniques used in data and information visualization. Yang et al. [26] proposed a visual hierarchical dimension reduction technique that creates meaningful lower dimensional spaces with representative dimensions from original data instead of from generated new dimensions. These techniques generate a lower dimensional subspace to reduce clutter but some information in the original data space is also lost. All these approaches unavoidably cause information loss, in one way or another.

Dimension ordering in visualization has been studied in [2, 25]. Ankerst et al. [2] proposed a method to arrange data dimensions according to the similarity between dimensions so that similar ones are put next 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. [25] imposed a hierarchical structure over the dimensions themselves, grouping a large number of dimensions into hierarchies 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. A visualization with the best correlated dimensions does not guarantee the least clutter. But the idea of using dimension ordering to improve clustering certainly inspired our work into the research of dimension ordering to improve visualization quality.

3 PARALLEL COORDINATES

Parallel coordinates is a technique pioneered in the 1980's that has been applied to a diverse set of multidimensional analysis problems [6]. 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 can be shown with very distinct shapes. In Figures 1 and 2, the two displays illustrate 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 very easy to see, but does not at all 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 polylines 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. These data points that don't belong to any cluster are called outliers. And we would want to minimize their impact in the display.

3.2 Clutter Measure in Parallel Coordinates

3.2.1 Clutter Definition

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 clutter 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:

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

Since $n-1$ and S_{total} are both constant, dimension orders that reduce the total number of outliers also reduce clutter in the display. In this way, we can measure the clutter in the display and then find the best order.

3.2.2 Algorithm for Computing 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 are only concerned with clusters within pairs of dimensions, we can use the normalized Euclidean distances between data points to measure their closeness. The two-dimensional clustering problem has been discussed intensely in the statistics, pattern recognition and data mining communities. Jain's book [7] gives a thorough description of clustering algorithms. Since our purpose is to find outliers that do not have any neighbors close to them, we decided to choose Lu and Fu [15]'s nearest-neighbor clustering algorithm. Suppose a set of data points $P = \{x_1, x_2, ..., x_n\}$ is to be partitioned into clusters. Let k denote the cluster number. The user specifies a threshold, t, on the nearest-neighbor distance. The algorithm can be described as follows:

- Step 1. Set $i \leftarrow 1$ and $k \leftarrow 1$. Take x_1 from P. Assign data point x_1 to cluster C_1.
- Step 2. Set $i \leftarrow i+1$. If x_i has not been assigned to any cluster, find the nearest neighbor of x_i among the data points already assigned to clusters. Suppose that the nearest neighbor is in cluster m. Let d_m denote the distance from x_i to this neighbor.
- Step 3. If $d_m \leq t$, then assign x_i to C_m. Otherwise, set $k \leftarrow k+1$ and assign x_i to a new cluster C_k.
- Step 4. If every point has been assigned to a cluster, stop. Else, go to step 2.

If a cluster contains only one data point, it is then called an outlier. In this way, we are able to find all the data points that

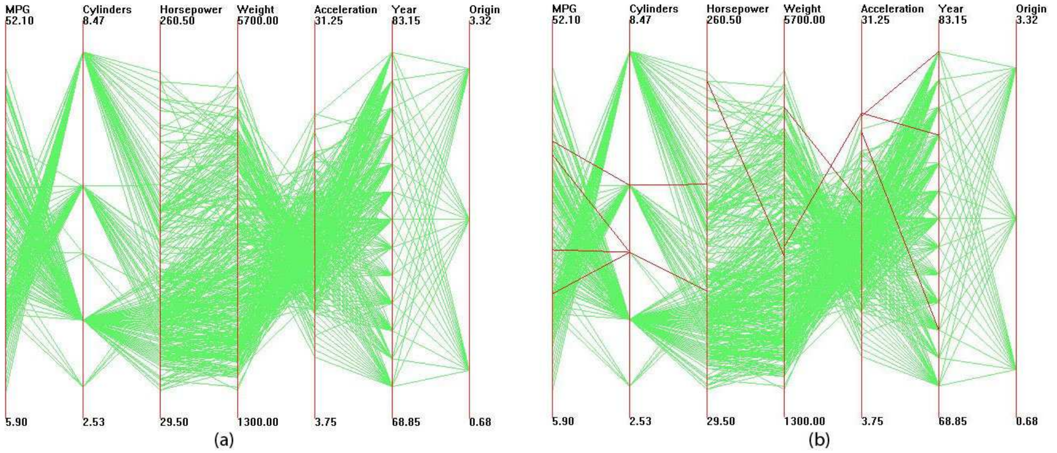


Figure 1: Parallel coordinates visualization of original Cars dataset. Outliers are highlighted with red in Fig.1-(b)

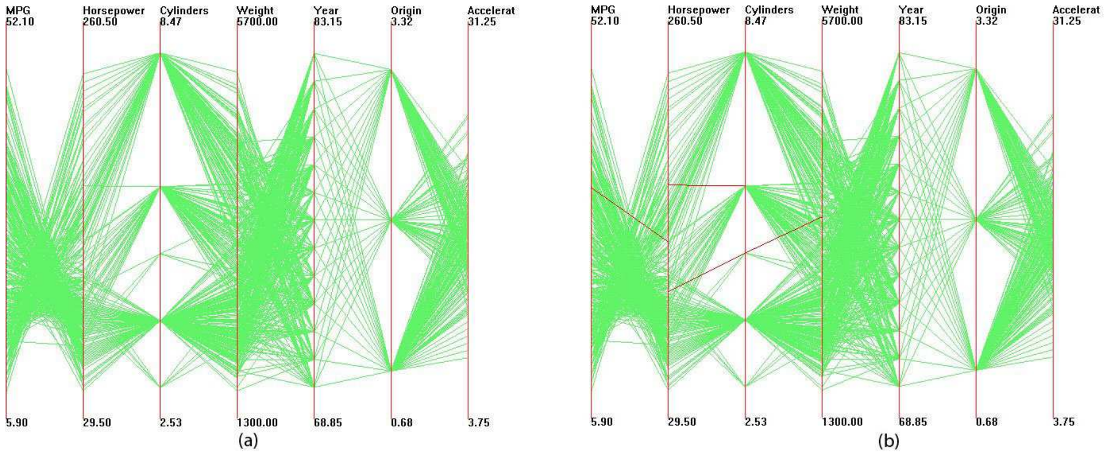


Figure 2: Parallel coordinates visualization of cars dataset after clutter-based dimension reordering. Outliers are highlighted with red in Fig.2-(b)

don't have any neighbors within the distance t in the specified two-dimensional space. We do this for every pair of dimensions and store the outlier numbers in a outlier matrix M. Given a dimension order, we can then count the total clutter by adding up outlier numbers between neighboring dimensions. If the dimension number is n, this is done in $O(n)$ time. Since the optimal dimension ordering algorithm is an exhaustive search algorithm with $O(n!)$ time, the search time involved is therefore $O(n*n!)$.

3.3 Examples

Figures 1 and 2 both represent the Cars dataset. In Figure 1 the data is displayed with the default dimension ordering. Figure 2 displays 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, the data points are better separated and 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 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. The data can also be rotated, isolated and masked to help the user to find structures that may not be visible in one of the simple orthogonal projections. However this manual projection pursuit approach is not efficient when dealing with high dimensional datasets. It is also likely to result in undetected structures since it's based on the user's knowledge and perception of the data. Automatic projection pursuit techniques [4] 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.

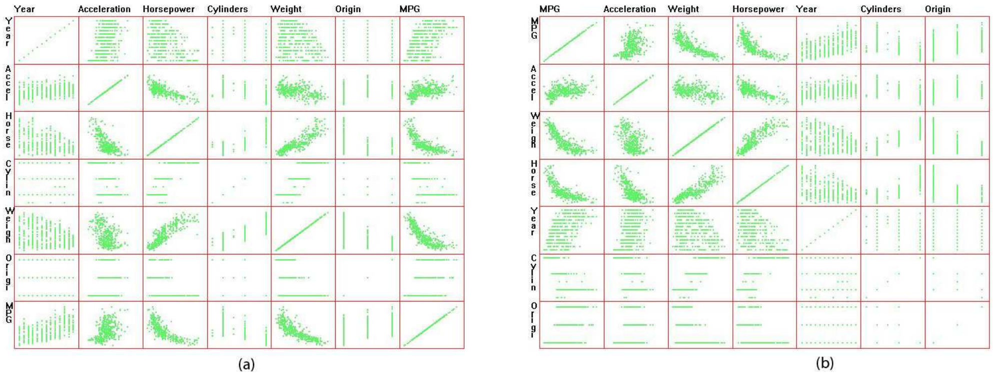


Figure 3: Scatterplot matrices visualization of Cars dataset. In Fig.3-(a) dimensions are randomly positioned. After clutter reduction Fig.3-(b) is generated. The first four dimensions are ordered with the high-cardinality dimension reordering approach, and the other three dimensions are ordered with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

With a matrix of scatterplots, users are not only able to find plots with structure, but also can view and compare the relationship between these plots. With scatterplot matrices visualizations, all two-dimensional plots are displayed on the screen. Thus 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 scatterplot matrices visualization. In these visualizations, we can separate the dimensions into two categories: high-cardinality dimensions and low-cardinality dimensions. In high-cardinality dimensions, data values are often continuous, 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 can only 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. However, a dimension being continuous or discrete does not inform us whether it has high or low cardinality. 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 N denote the total number of data points. If $m_i \geq N$, dimension i is considered high-cardinality, otherwise it is low-cardinality.

We will treat high-cardinality dimensions 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 Clutter Definition

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:

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

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 are symmetric along the diagonal), and reorder the dimensions so that similar plots are displayed as close to each other as possible. We will 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 $\sqrt{(1-0)^2 + (1-0)^2} = \sqrt{2}$. The larger this number is for a display, the more cluttered it is. We then define the total distances between similar plots to be the clutter measure.

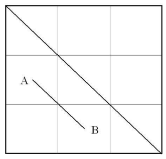


Figure 4: Illustration of distance calculation in scatterplot matrices.

4.2.2 Algorithm for Computing Clutter

In high-cardinality dimension space, the 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.

- Step 1. A correlation matrix $M(n,n)$ is generated for all n dimensions. $M[i][j]$ represents the Pearson correlation coefficient for the plot on the ith row, jth column.

- Step 2. $i \leftarrow 0$. Visit plot p_0. Find all the unvisited plots that have similar correlation with p_0, i.e., the differences between their Pearson correlation with p_0's are within threshold t. Calculate their distances from p_0 on the display, and add them to the total clutter measure.

- Step 3. $i \leftarrow i+1$. Visit p_i. Find all unvisited plots similar enough to p_i. Calculate their distances from p_i on the display, and add them to the total clutter measure.

- Step 4. If all plots have been visited, stop. Otherwise go to step 3.

This way, we will get a total distance for any scatterplot matrices display. With this measure, we will be able to make comparisons between different displays of the same data. Unlike the one-dimensional parallel coordinates display, we have to calculate distance for every pair of plots. This is a $O(n^2)$ process. We will do the exhaustive search for best ordering, so the total computing time will be $O(n^2 * n!)$.

4.3 Low-Cardinality Clutter Measure in Scatterplot Matrices

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

Plots involving low-cardinality dimensions are very different in display pattern from those only involving high-cardinality dimensions. The user's perception will naturally envision them as two different types of patterns.

For plots with low-cardinality dimensions, the higher the cardinality, the more crowded the plot seems to be. Therefore, we have a different measure of clutter for these dimensions. 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.

With low-cardinality dimensions, the dimension reordering can be envisioned as a sorting problem. With a quick sort algorithm, we can then achieve it 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 will be useful in helping the user identify similar low-cardinality dimensions and find similar plots more easily 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 [14]. XmdvTool 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 then 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. 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. Suppose we want to find structure in one glyph. For this glyph, we may call it 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. Then in a perfectly structured glyph we have:

- 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) make them more symmetric and easy to interpret.

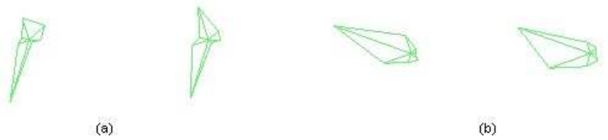


Figure 5: The two glyphs in Fig.5-(a) represent the same data points as Fig.5-(b), with a different dimension order.

5.2 Clutter Measure in Star Glyphs

5.2.1 Clutter Definition

To reduce the clutter for the whole display, we seek to reorder the dimensions for the purpose of minimizing 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.

5.2.2 Algorithm for Computing Clutter

In order to calculate clutter in one display, we test every glyph for its monotonicity and symmetry. Suppose the user chooses monotonically increasing and symmetry as the structure measure. 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. If for a point's normalized values on two neighboring dimensions ($dimension_{n-1}$ and $dimension_0$ are not considered neighbors) p_i and p_{i+1}, p_{i+1} is less than p_i, we will see if $p_i - p_{i+1}$ is less than threshold t1 or not. If so, we consider this non-monotonicity occurance as tolerable. If not, we will add this occurance 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.

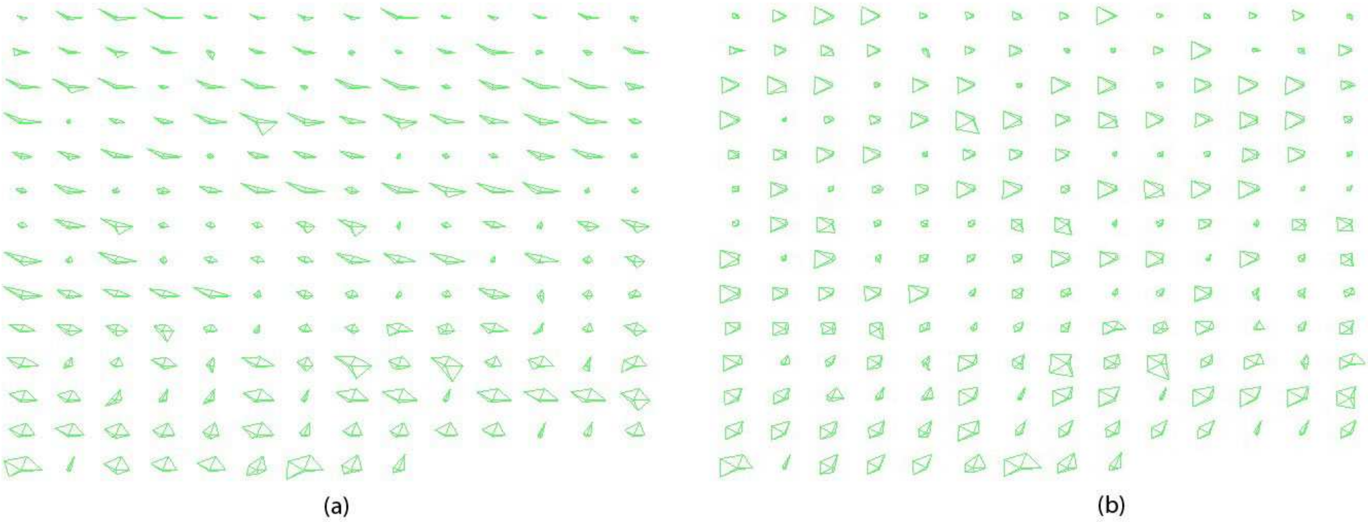


Figure 6: Star glyph visualizations of coal disaster dataset. Fig.6-(a) represents the data with original dimension order, and Fig.6-(b) shows the data after clutter being reduced.

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 then $O(n*m*n!)$.

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 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 projection method developed by LeBlanc et al. [12]. Each dimension of the dataset is first discretized into a user-specified 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.

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 outer-most 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 can reveal much more information. 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 as 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 Clutter Definition

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 which minimizes this number will then be considered the best order. Besides that, we need to also define which bins are considered neighbors. The choices are 4-connected bins and 8-connected bins. With 4-connected neighbors, the points considered aggregated will share the same data range on all but one dimension, while the 8-connected bins may fall into different data ranges on at most two dimensions. And since they are connected, their values on those dimensions have to fall into immediately neighboring value ranges.

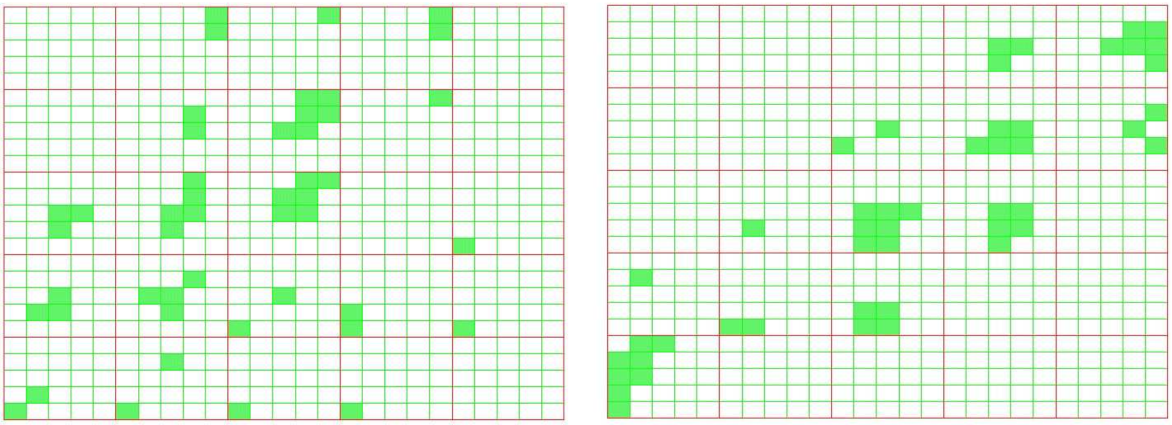
6.2.2 Algorithm of Computing Clutter

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.

The 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, and the optimal search will take $O(m^2*n!)$.

6.3 Example

An example of clutter reduction in dimensional stacking is given in Figure 7. We have defined 8-connected as our measure for neighbor.



(a) (b)

Figure 7: Dimensional stacking visualization for Iris dataset. Fig.7-(a) represents the data with original dataset, and Fig.7-(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	3secs
		Cereal-Part	77	10	23secs
		Voy-Part	744	11	4:02mins
Scatterplot Matrices	$O(n^2*n!)$	Voy-Part	744	11(6 high-card dimensions)	5 secs
		AAUP-Part	1161	9	3:13mins
Star Glyphs	$O(m*n*n!)$	Cars	392	7	18secs
Dimensional Stacking	$O(m^2*n!)$	Coal Disaster	191	5	10secs
		Detroit	13	7	2:10mins

Fig.7-(a), denoting the original data order, is composed of many ”islands”, namely the filled bins without any neighbors to them. In Fig.7-(b), the optimal ordering, there are much fewer ”islands”, resulting in an easier interpretation.

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.

Ideally, we would hope to use an exhaustive search to find a 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 Traveling Salesman Problem. Therefore, we can do the optimal search with only low dimensionality datasets. To get a quantitative understanding of this issue, we did a few experiments for different visualizations, and the results we obtained are presented in table 1. We realized that even in a low dimensional data space - around 10 dimensions - the computational overhead can 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 by us.

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 we will leave the old arrangement intact and go on swapping another pair of dimensions. Keep doing this a number of times 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 are added into the tour. The greedy algorithm [3] keeps adding the nearest possible pairs of dimensions into the tour, until all the dimensions are in the tour.

The nearest-neighbor and greedy algorithms are good for parallel coordinates and scatterplot matrices displays, because 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 don’t have a direct measure of dimension relationship. Thus these algorithms are not very amenable to the latter two techniques.

With heuristic algorithms, we can work on dimension reordering with much higher dimensions with relatively good results. Experimental results are presented in Table 2.

8 Conclusion and Future Work

In this paper, we have proposed the concept of visual clutter 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 then used reordering algorithms to search

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	2secs
				Greedy Algorithm	3secs
				Random Swapping	2secs
	AAUP	1161	14	Nearest-Neighbor Algorithm	7secs
				Greedy Algorithm	9secs
				Random Swapping	6secs
Scatterplot Matrices	Census-Income	200	42	Nearest-Neighbor Algorithm	2secs
				Greedy Algorithm	3secs
				Random Swapping	2secs
	AAUP	1161	14	Nearest-Neighbor Algorithm	8secs
				Greedy Algorithm	8secs
				Random Swapping	7secs
Star Glyphs	Census-Income	200	42	Random Swapping	2secs
	AAUP	1161	14	Random Swapping	7secs
Dimensional Stacking	Those datasets are too big for dimensional stacking visualization.				

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. In this paper, we only discussed the usage of dimension order for reducing clutter. However, there are certainly other visual aspects that affect clutter or structure in a display and thus can help facilitate the interpretation of a visualization.

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 Symposium on Information Visualization*, pages 52–60, 1998.

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

[4] 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.

[5] 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.

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

[7] Anil K. Jain and Richard C. Dubes. *Algorithms for clustering data*. Prentice-Hall, Inc., 1988.

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

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

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

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

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

[13] 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.

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

[15] S. Y. Lu and K. S. Fu. A sentence-to-sentence clustering procedure for pattern analysis. *IEEE Transactions on Systems, Man and Cybernetics*, 8:381–389, 1978.

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

[22] G.J. Wills. An interactive view for hierarchical clustering. *Proc. IEEE Symposium on Information Visualization*, pages 26–31, 1998.

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

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

[25] 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.

[26] 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.