Revealing structure in visualizations of dense 2D and 3D parallel coordinates

Jimmy Johansson*, Patric Ljung, Mikael Jern, Matthew Cooper

Norrköping Visualization and Interaction Studio, Linköping University, Sweden

Abstract

Parallel coordinates is a well-known technique used for visualization of multivariate data. When the size of the data sets increases the parallel coordinates display results in an image far too cluttered to perceive any structure. We tackle this problem by constructing high-precision textures to represent the data. By using transfer functions that operate on the high-precision textures, it is possible to highlight different aspects of the entire data set or clusters of the data. Our methods are implemented in both standard 2D parallel coordinates and 3D multi-relational parallel coordinates. Furthermore, when visualizing a larger number of clusters, a technique called 'feature animation' may be used as guidance by presenting various cluster statistics. A case study is also performed to illustrate the analysis process when analysing large multivariate data sets using our proposed techniques.

Keywords

Parallel coordinates; 3D multi-relational parallel coordinates; clustering; transfer function; density map; feature animation

Introduction

In various disciplines (such as economics, control theory, physics and biology) multivariate data sets are becoming more and more common. Depending on the discipline and the a-priori knowledge of the data, different analysis processes must be performed. In the end, however, a visualization of the data is often extremely useful. When it comes to the visualization of multivariate data, parallel coordinates [1, 2] is widely used and is the technique we have chosen to build upon. Parallel coordinates is used to transform multi-dimensional data items down to a 2D display with parallel axes. Each N-dimensional data item is represented as a polyline intersecting all axes. Despite the popularity of parallel coordinates, the technique has two main limitations:

- Even for a medium sized multivariate data set the display suffers from 'over-plotting', resulting in an image which is far too cluttered to perceive any trends, anomalies or structure.
- The parallel axes configuration prevents correlation analysis between all but adjacent axes.

These limitations have been separately addressed in two of our previous papers [3, 4]. In this paper, we revisit the fundamentals of those papers. In [3] methods for revealing and displaying cluster structure based on using high-precision structure textures and outlier textures are developed and these methods treat all data items in a cluster as a single image making interactive analysis of large data sets possible. Transfer Functions (TFs) are then used on the high-precision textures to reveal different aspects of the clusters. A technique called 'feature animation' is also introduced and is used to present cluster properties. We now extend our high-precision texture and TF techniques from working only on subsets of the data to include the analysis of unclustered data. We also propose a technique to ensure a consistent colour coding of the clusters. The 3D multi-relational parallel coordinates technique

* E-mail: jimjo@itn.liu.se

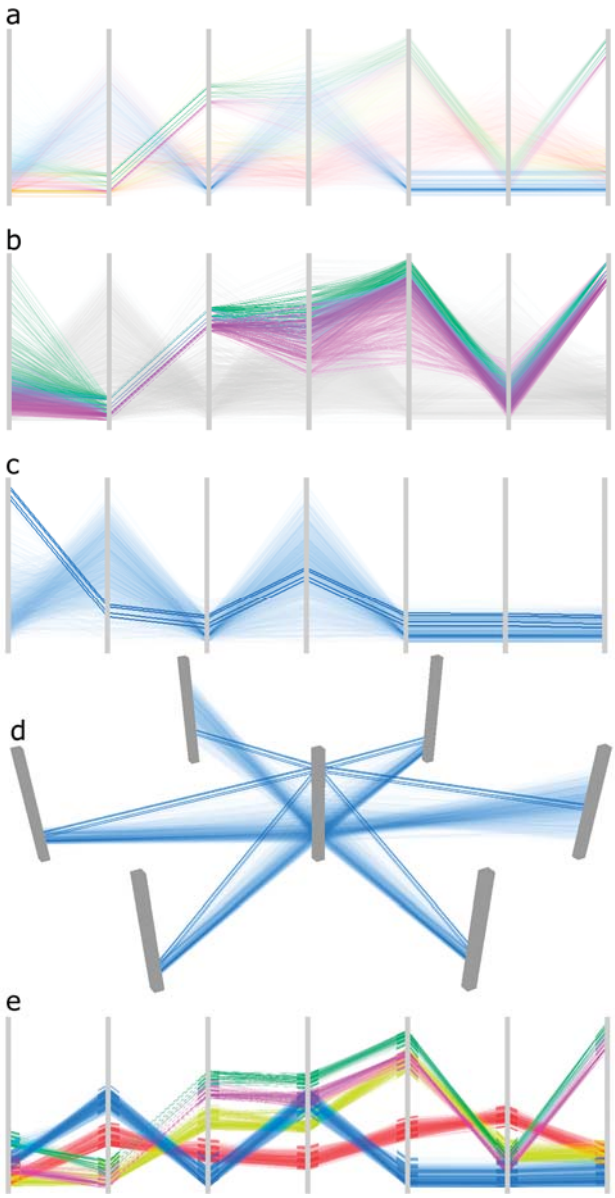

Figure 1: Five clusters visualized. TFs map density values (representing the degree of line overlap) to opacity. (a) All clusters visualized using a linear TF. (b) two similar clusters are selected for a more detailed analysis. A square root TF emphasizes low-density regions. (c) Potential local cluster outliers are enhanced. (d) The same outliers as in (c) are highlighted in 3D multi-relational parallel coordinates. (e) Clusters are displayed as uniform bands with 'feature animation' enabled.

described in [4] employs clustering but is limited by cluttering when it comes to the display of large multivariate data sets. Therefore, we incorporate the high-precision texture, TF and outlier techniques into this visualization. We also use linked views so that different views of the same data can easily be accessed, a change in one view instantly updating the other view. 2D parallel coordinates gives a good overview of the data and can be used to set the focus dimension for 3D multi-relational parallel coordinates. A summary of our proposed techniques is illustrated in Figure 1.

This paper also presents a case study where we illustrate how our techniques can be simultaneously used in an analysis process to analyse large multivariate data sets and show the importance of interactive cluster structure mapping through TFs. For the purpose of our clustering experiments, we have chosen to use the well-known K-means algorithm [5, 6] to classify data items into exclusive clusters although other clustering techniques may be equally usable.

After presenting previous work, this paper describes our methods for creating high-precision textures and how TFs can be used in conjunction with parallel coordinates. A method for analysing local cluster outliers and our feature animation technique are then presented. After that, we describe how our techniques are implemented in 3D multi-relational parallel coordinates. The next section deals with performance and implementation details. Finally, we present a case study of how our techniques may be used to analyse a multivariate data set and discuss our results and planned future research work.

Related work

Over the past decades, extensive research concerning parallel coordinates has been done. In general, this work can be divided into research aiming to refine 2D parallel coordinates and research about extending the method to higher dimensions. When it comes to refining the 2D parallel coordinates technique, most work has been concerned with the limitation of displaying large data sets.

Clutter reduction approaches

As early as in 1987, Hinterberger [7] used data density as an abstraction to visualize multivariate data using parallel coordinates. More recently, Fua et al. [8] proposed a multiresolution view of the data via hierarchical clustering which lets the user navigate the resulting structure to locate a particular level of detail. Berthold and Hall [9] used parallel coordinates, incorporating membership information, to visualize fuzzy clusters. "Striped" envelopes and ellipse plots were introduced by Andrienko and Andrienko [10] as two methods for displaying properties and structure of subsets in parallel coordinates. Another approach based on the concept of representing each cluster as an envelope or polygon was presented by Novotny [11]. While several of these techniques display a good summarization of the data, they are less successful in revealing important structure information.

Another way to tackle the problem of cluttered parallel coordinates displays was suggested by Miller and Wegman [12], where line density plots are used. Density plots were also implemented by Wegman and Luo [13], where each line is rendered with a user-defined opacity value aiding in the visual search for structure. A different approach to facilitate the analysis of cluttered parallel coordinates was developed by Rodrigues et al. [14]. Their technique is based on frequency information and is used to highlight highly populated regions in cluttered parallel coordinates. The same problem has also recently been addressed by Artero et al. [15] by means of frequency and density plots. Most of these techniques are based on linear density mappings and, when analysing large data sets, precision and saturation problems tend to occur, causing structure to be lost.

3D parallel coordinates

In adding a 3rd dimension to the parallel coordinates display, the goal is to make as good use as possible of the new dimension so that a more efficient analysis can be performed. 3D parallel coordinates has received some attention during the last year and a number of extensions have been proposed. Fanea et al. [16] combine parallel coordinates and star glyphs into a 3D representation. This technique, while possibly reducing some of the cluttering by unfolding the parallel coordinates display, makes it difficult to perceive correlation. Extending standard parallel coordinates to 3D has also been addressed by Johansson et al. [4] where a clustered 3D multi-relational parallel coordinates technique was proposed. This visualization technique allows a simultaneous one-to-one relation analysis between a focus dimension and the other dimensions.

Representing structure information with high-precision textures

The lines representing the different data items in the parallel coordinates display overlap and cross each other. This is what makes trends and structure visible and this information is important for the analysis. As the number of data items increases, however, it becomes more and more difficult to preserve this information due to cluttering artefacts. As proposed in [13], a density map can be

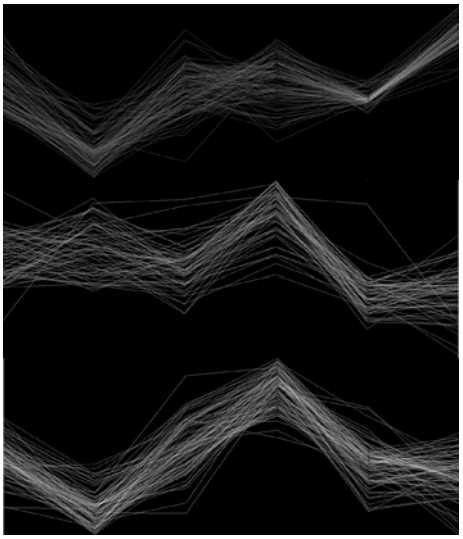

Figure 2: The high-precision structure texture which, applied to a coloured polygon, produces the final visual display of the clusters. This example shows information about three clusters for 2D parallel coordinates.

generated by using opacity and additively blending lines making the structure more easily seen. This is, however, not sufficient to reveal the structure in large data sets due to the limited precision of graphics cards and monitors. This limited precision will cause the smallest opacity values to disappear and structure to be lost. Also, when dealing with very large data sets, the number of lines to be rendered may cause limitations on interactivity. As an alternative to rendering each individual line, high-precision textures will be used to generate a density map that later can be used as a basis for opacity mapping. The high-precision texture approach can be used to construct density maps both for individual clusters as well as for the entire data set.

High-precision texture generation

A high-precision texture can be defined as

$$T = t_{m,n} \in \mathbf{N}, \text{ where } t_{m,n} < 2^B \text{ and } B > 8. \quad (1)$$

Typically $B = 16$ or $B = 32$ is used. **N** denotes the natural numbers. The high-precision textures to hold the structure information are created in a pre-processing step using graphics hardware. This is a fast operation but it introduces a precision problem due to the limited precision of the framebuffer. A framebuffer having B bits per component supports 2^B different levels (in our case we use these levels to represent density values by means of grey scale intensity values). On standard graphics hardware B is usually 8, yielding 256 different levels. If a data set displayed with parallel coordinates contains more than $M = 2^B - 1$ different data items, all information may not be revealed using additive blending. This can be resolved by rendering the data in subsets of M items and accumulating the results in a higher precision buffer. This ensures that the maximum number of density levels in each subset never exceeds the precision of the framebuffer.

In order to ensure that the maximum available density range is used, it is necessary to count the number of intersecting lines present anywhere in the parallel coordinates display, not simply those which occur at the axes. Using a texture this is trivial by using a max operator on the texels. The maximum density value, ρ, is used to normalize the range. The complete procedure for creating high-precision structure textures for 2D parallel coordinates is outlined below; see Figure 2 for an example of the result. If no clustering is used, $C = 1$. C is the number of clusters.

1. Specify a drawing area to be used for rendering lines in the framebuffer and a width and height for a 2D texture.
2. For cluster *i*, where $i = 1, \ldots, C$, perform the following steps:
    a. Select all data items belonging to cluster *i*.
    b. Render lines for the selected data items in subsets of *M* data items. The density value for each line is set to ε, the smallest value given the available precision in the framebuffer. Use additive blending to combine the contributions of the individual lines. After each rendered subset, read pixel values from the framebuffer and accumulate the result.
    c. In the resulting image (containing the accumulated results) find the maximum density value ρ_i.
3. Find ρ_{global}, the largest of the ρ_i values and normalize the density ranges. Depending on the normalization method used, different aspects are revealed. If we normalize the resulting images using ρ_{global}, clusters with a small maximum intersecting value will be more transparent. Normalizing each cluster individually using ρ_i, each cluster's maximum intersecting value will be perceived as equally dense. Both approaches are equally viable but highlight different aspects.

Cluster shapes and colouring

If parallel coordinates is used to display clusters, each cluster can either be represented as a uniform band, displayed at the mean value of the cluster, or represented with its true size. Evidently, the latter representation gives a correct visual mapping but representing each cluster as a uniform band may, in some situations, have its advantages. First of all, for a larger number of clusters, a thin band will not occupy as much vertical space and can therefore be used to give a good overview of the data. The uniform bands are drawn with a relative width, ψ, according to $\psi = \frac{K}{K_{\max}}\kappa$, where *K* is the population of the current cluster, K_{max} is the population of the largest cluster and κ is a user-defined scaling factor which can be changed during run-time. It is also possible to interactively switch between the uniform band and true size cluster representations, as well as to select one or more clusters for a more detailed analysis.

The hue, saturation and value (HSV) colour model [17] is used to assign cluster colours. The saturation and value components are set to fixed values and the angle, ϕ, between each hue component is calculated as $\phi = \frac{2\pi}{C}$. This gives a good colour separation between the clusters and they are easy to visually distinguish. However, since there is no specific order by which some clustering algorithms (including K-means) construct the clusters, the colours assigned to each cluster may be different between different runs. We use a simple consistent colouring technique to deal with this problem by assigning the colours based on specific cluster properties. A simple but effective property to use is the mean value. The colours for the clusters are assigned based on their order when sorted by their mean values, along a specific dimension which can be user-defined or automatically selected. The proposed procedure for automatic selection uses a measure of spread between the ordered cluster mean values:

$$\xi = \frac{\sum_{i=1}^{C}(\mu_i - \hat{\mu}_i)^2}{\mu_C - \mu_1}. \qquad (2)$$

The value ξ is calculated for each dimension independently and the most suitable dimension is the one satisfying $\underset{j}{\operatorname{argmin}}(\xi_j)$. μ_i is the mean value of cluster *i* and $\hat{\mu}_i$ is a predicted mean value giving the largest possible spread. μ_C is the largest mean value and μ_1 the smallest.

Interactive analysis using transfer functions

The high-precision textures ensure that no structure information is lost during pre-processing. The next step is now to map these density values to opacity to form the final visual result. As in previous work [15], a linear function may be used to control how density values are mapped. However, this has the limitation, when emphasizing low-density regions, that higher-density regions will be saturated and so a major part of the structure could be lost. Even without this limitation, linear mapping is undesirable due to perceptual issues. To tackle this problem, density values can be mapped using non-linear and user-defined mappings through Transfer Functions (TFs). Since TFs can be of arbitrary shape they can be used to focus on any specific density range. A TF can be defined as

$$T(s) = \alpha(s), \text{ where } \alpha(s) \in [0, 1] \text{ and } s \in [0, \rho\,] \qquad (3)$$

The opacity value stretches from completely transparent (0) to fully opaque (1).

A number of pre-defined TFs may be used to display different properties of the entire data set or clusters. Figure 3 illustrates how the visual appearance of a single cluster changes with different TFs. Figure 3(a) shows the result of applying a linear TF. A quadratic TF (Figure 3(b)) filters out all low-density regions leaving only high-density regions visible. If the data set contains much noise, this TF may help clear up the view. In contrast, in order to enhance low-density regions, a square root or logarithmic TF can be used (Figures 3(c) and 3(d)).

Besides using a pre-defined TF, it is possible to draw a function either free-hand or by assigning a number of control points. This gives the user an active role in the visualization process and provides a more powerful analysis than by using only predefined TFs. Since the TFs operate on textures, the feedback is instantaneous and the response time is the same regardless of the size of the data set or number of clusters. Figure 4 shows an example of the interactive TF editor. When drawing a free-hand TF, one common thing to do is to try to isolate low-density regions in order to search for anomalies. This can, however, be difficult due to the high-density range: the precision in the low region of the drawing space is simply not high enough. Instead of using a linear drawing space it is more appropriate to draw the TF using a square root or logarithmic space. This gives the necessary drawing precision and it is easier to isolate low-density regions. In addition, when analysing low-density regions, mapping density values of 0 to an opacity value of 0 can be enforced.

Enhancing potential outliers

Outliers play an important role when investigating a data set. In the most general case an outlier is a data item that differs from the main trend in the data set. This can be measured in many different ways, see [18] for a more detailed discussion. In our case we focus on local outliers, a data item that differs from the main trend in a particular cluster. We have used a method based on the interquartile range [19] to define if a data item is an outlier or not but, depending on the characteristics of the data set, other methods may be more appropriate. For each cluster, *i*, and dimension, *j*, the interquartile range, $Q_{i,j}^{IQR}$, is defined as the difference between upper and lower quartiles, $Q_{i,j}^{3} - Q_{i,j}^{1}$, where $Q_{i,j}^{1}$ is the 25th percentile and $Q_{i,j}^{3}$ is the 75th percentile. The data items that belong to cluster *i* are determined to be outliers if they fall $\beta Q_{i,j}^{IQR}$ above $Q_{i,j}^{3}$ or below $Q_{i,j}^{1}$. This is done for each dimension, *j*, independently. The value of β may be interactively changed but one commonly used value, when searching for outliers, is to use β = 1.5. Larger values of β can be used to identify more extreme outliers.

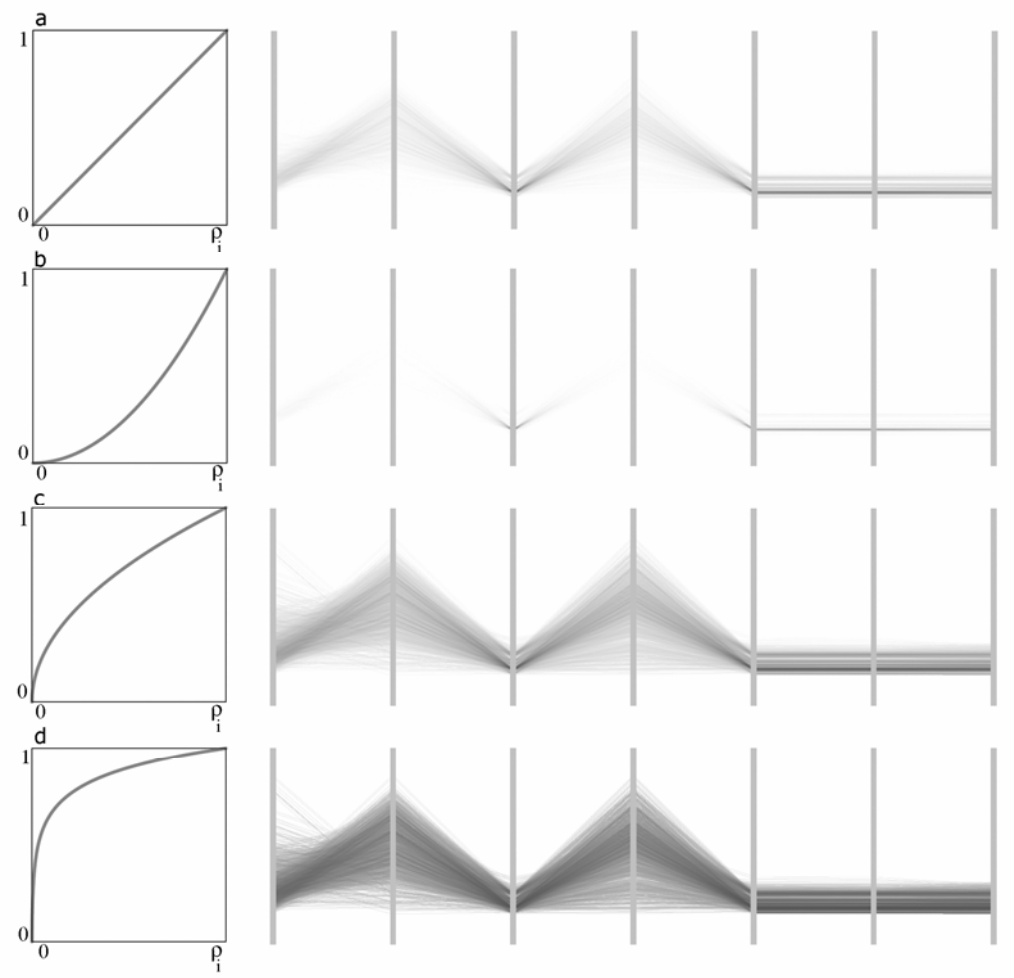

Figure 3: Four TFs are used to display different properties of a cluster visualized with parallel coordinates. In this example, the maximum line overlap, ρ_i is 1023. (a) The high-precision structure texture with a linear TF applied. (b) A quadratic TF is used to filter out all but high-density regions. (c) A square root TF gives the opposite result compared to (b), emphasis being put on low-density regions. (d) The logarithmic TF further enhances low-density regions.

Since the outlier test is applied to each dimension independently, the matrix containing the data items classified as outliers, R, may contain doubles or triples and so on. Having the same data item twice in *R* means that the data item is an outlier in 2 dimensions. A threshold value, γ, may be used to specify in how many dimensions a data item needs to be classified as an outlier in order to be deemed to be an outlier. The complete procedure for constructing the outlier texture is as follows.

1. Specify a drawing area to be used for rendering lines in the framebuffer and a width and height for a 2D texture.
2. For cluster *i*, where $i = 1, \ldots, C$, do the following steps:
    a. Use the interquartile range to test if the selected data items for cluster *i* are potential outliers.
    b. In matrix *R*, find the data items that appear with a frequency greater or equal to γ.
    c. Render lines for the data items found in step b with a user-defined opacity value and read pixels from the framebuffer.

The enhanced outliers are displayed using a separate texture, thus they can be easily switched on or off. By studying the enhanced outliers in Figure 5 we get information about the performance of the

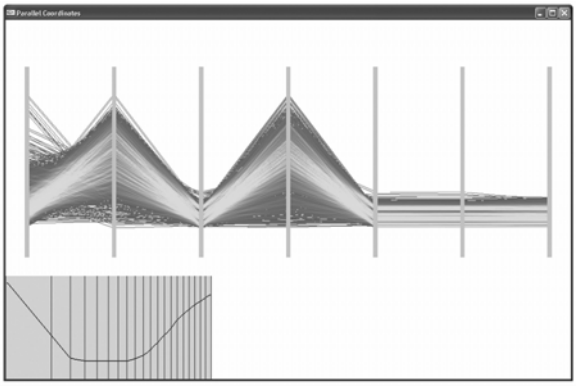

Figure 4: A TF has been hand-drawn, in a square root space.

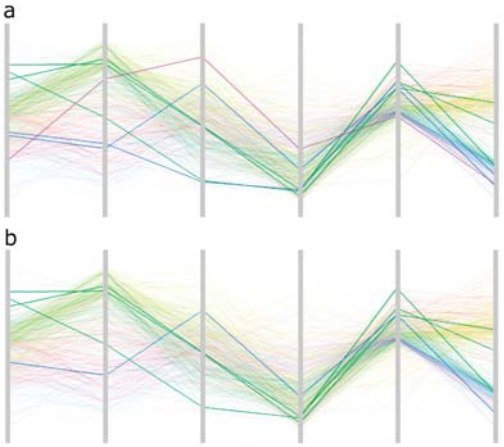

Figure 5: When analysing several clusters simultaneously, local outliers can be enhanced and tracked through high-density regions. (a) Enhancing the six strongest potential outliers (β = 1.5, γ = 2). (b) Enhancing the four strongest potential outliers (β = 2.5, γ = 2).

clustering algorithm used. In Figure 5(b) it is seen that the green and blue clusters have the most distinct outliers. Figure 5(a) also shows that the magenta cluster contains a strong outlier. The same visualization concept as described here could also be used to highlight global outliers, data items that differ from the main trend of the entire data set, although other identification techniques than the interquartile range would need to be applied.

Feature animation

Animation has been widely used in visualization, often to enhance the understanding of time-varying variables. In conjunction with parallel coordinates, however, there are few examples. Barlow and Stuart [20] animated the parallel coordinates display to enhance the understanding of how objects within the multi-dimensional space change over time. We also use animation in the parallel coordinates display but for a different purpose: we do not use it for animating time changes but instead try to convey statistical properties about clusters. This technique, which we call ‘feature animation’, is used to disclose the skewness, standard deviation or variance of each dimension and cluster and consists of a moving pattern of lines, with different phase velocities, rendered close to each axis. The lines are implemented using a pre-generated texture and the appearance of movement is created by translating the polygon's texture coordinates.

For the standard deviation and the variance the phase velocity, $v_{i,j}$, for cluster i and dimension j is always positive, while for the skewness it is possible to display positive as well as negative values. The animation is performed by using $v_{i,j}$ to determine the speed with which the vertical texture coordinates should be translated. By animating the standard deviation or variance the user can quickly examine each cluster to see if the cluster is tight or loose. Typically, we want to isolate loose clusters and try to find out why the clustering algorithm has done a poor job of constructing these clusters. The skewness is a measure of the amount of asymmetry in the distribution. The direction and speed of the animation indicate the direction and magnitude of the skewness, respectively. More details can be found in [3]. Figure 6 shows a static illustration of the feature animation, with the moving lines close to each axis, applied to 7,800 7-dimensional data items classified into 10 clusters. This gives 70 different intersection points that may require analysis. In the same way as the outlier texture, this texture may be independently switched on or off.

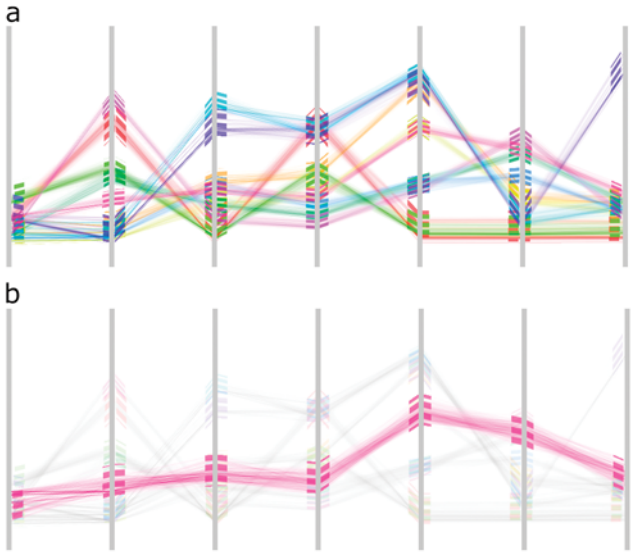

Figure 6: 'Feature animation' is used to present cluster statistics to the user. (a) An overview of all clusters displayed as uniform bands with feature animation enabled. (b) A single cluster is selected for more detailed study.

3D multi-relational parallel coordinates

Our proposed techniques have so far only been discussed in conjunction with 2D parallel coordinates. In this section we extend these techniques to also apply to 3D multi-relational parallel coordinates.

In contrast with 2D parallel coordinates, where all axes are mapped to a plane, the 3D multi-relational parallel coordinates technique has the axes mapped onto a cylinder with a focus axis at its centre (one example is given in Figure 1(d)). The advantage of this technique is that it allows a simultaneous one-to-one relation analysis between the focus axis and all other axes. Any of the outer axes can, with a simple mouse click, be made the focus axis and this allows a much more controlled correlation analysis compared to 2D parallel coordinates. Since we also link the two techniques together, they can be simultaneously used to present different views of the same data. The choice of focus dimension can also be made by clicking on an axis in the 2D parallel coordinates display. When constructing the high-precision textures for 3D multi-relational parallel coordinates, steps 2 and 3 of the procedure for creating high-precision textures for 2D parallel coordinates need to be modified according to:

2. For cluster i, where $i = 1, \ldots, C$, perform the following steps:
    a. Select all data items belonging to cluster i.
    b. Render lines for all relations, ζ, between the focus dimension and the other dimensions of the selected data items in subsets of M data items. $\zeta = 1, \ldots, N - 1$, where N is the number of dimensions of the data set. The density value for each line is set to ε. After each subset is rendered, read pixel values from the framebuffer and accumulate the result. This gives a separate image for each pairwise combination between the focus dimension and the other dimensions.
    c. In the resulting $N - 1$ images (containing the accumulated results), find the maximum values, $\rho_{i,\zeta}$.
3. Find ρ_i, the largest of the $\rho_{i,\zeta}$ values and ρ_{global}, the largest of the ρ_i values and normalize the density range.

When constructing the outlier texture, the same changes as described above in step 2(b) also need to be done for step 2(c) described in the section on potential outliers. That is, instead of rendering a

single image for each cluster, images for each relation, ζ, between the focus axis and the other dimensions need to be constructed.

Performance and implementation

Our visualization methods are implemented using C++ and OpenGL. The time it takes to create the high-precision textures in the pre-processing step depends on the texture size, the number of data items and the dimensionality of the data set. As an example, using a standard desktop PC with a 2.4 GHz Intel P4 CPU, 1 GB RAM and an NVIDIA Quadro FX 500 graphics card, it takes approximately 700 ms to create high-precision textures for 10 clusters of a 7-dimensional data set containing 7,800 data items. Each cluster representation is constructed using a texture of size 1024 × 512. Applying TFs to the high-precision textures is even less computationally expensive and the same 10 clusters are updated in approximately 90 ms. For the logarithmic TF, which is by far the most time-consuming operation, a lookup table was used. If an extremely large texture is used, the delay when applying a TF could become a problem. This can, however, be overcome by using a floating-point buffer and performing all operations directly on the GPU. During the visualization, we are able to maintain a frame-rate of 140 frames per second, using a display window with a resolution of 1600 × 800 pixels, regardless of data size. The K-means algorithm, when applied to a data set containing 17,000 6-dimensional data items, takes less than one second to partition the data into five clusters.

One limitation of our texture approach is that selecting a single data item or a subset of data items is more complicated than when drawing each line individually. This is because all data items are treated as the same object. A seemingly straightforward solution for this would be to compare the coordinates of each data item and the mouse pointer. However, when visualizing a large number of data items, the pixel resolution of the mouse pointer would not be high enough since the width of each line would be smaller than a single pixel. The same problem would, however, be present when rendering each data item as an individual line. Currently, these types of selections within the clusters are not supported in our system.

Case study

In this section, we conduct a case study on a synthetic data set [21] containing 17,000 6-dimensional data items with five clusters. The five clusters contain 5,000, 4,000, 3,000, 2,500 and 2,500 data items, respectively, and all have Gaussian distributions. The clusters are deliberately constructed so that they have extensive overlap in several dimensions but can still be separated. We give an illustrative example of how an analysis process can be performed using our techniques and, for this experiment, the K-means clustering algorithm is used to classify the data items into exclusive clusters. Our visualization methods can be used to analyse the results from a large variety of clustering algorithms, the entire data set without clusters or the result from a user selection. Since the K-means algorithm randomly order the identified clusters, our consistent colouring technique is used to remove the colour variation.

With our analysis we would like to answer the following questions:

1. Is there any structure at all in the data set?
2. How many clusters do we need to best describe the relations within the data set?
3. Is it possible to see if this is an appropriate number of clusters?

Looking for structure

Figure 7(a) shows a visualization of the data set using standard unmodified parallel coordinates. As can clearly be seen, we cannot draw any conclusions at all from this image. Using high-precision textures and a simple linear TF does, however, reveal something. Figure 7(b) indicates that some structure exists in the data. Using 3D multi-relational parallel coordinates, we can also quickly browse through and examine the correlation between all different dimension combinations. Figure 7(c) shows an example where the leftmost dimension in the 2D parallel coordinates display, Figure 7(b), has been

chosen as the focus dimension. This convinces us that there is enough structure in the data set to continue the analysis. From Figures 7(b) and 7(c), we can even make a qualified guess that there are at least four clusters but we cannot be sure if there are more.

Determining the number of clusters

In this case we could make a qualified guess about the number of clusters in the data set so we start by classifying the data items into four clusters. If this was not the case, we could start by constructing two clusters and keep adding clusters until we are satisfied. In this case we start by illustrating the analysis procedure from four clusters. Since each cluster is constructed using a separate texture, it is easy to browse through all clusters. Figures 8(a–d) show these four clusters visualized using a linear TF applied to the high-precision textures. This TF gives a good overview of the clusters but important information may be lost due to the large range of density values. By, for example, studying Figures 8(a) and 8(d) we get an indication about that they are not completely homogeneous. To further investigate the clusters we visualize them using a logarithmic TF that puts more emphasis on low-density regions, Figures 8(e–h). The clusters shown in Figures 8(e) and 8(h) clearly indicates a suspicious spread in several dimensions and the cluster shown in 8(h) also has a hole in it (2nd dimension). Furthermore, the cluster shown in Figure 8(g) has a number of outliers. To be absolutely sure that we still have not missed any information, we draw a TF by hand to increase the low-density regions as much as possible. Though this clutters the structure in the high-density regions, it is in this particular case useful. This analysis indicates that we have not used the most appropriate number of clusters and hence, the next step would be to increase to five clusters.

The result of using five clusters is illustrated in Figures 9(a–e). In this example, we have visualized our clusters using a logarithmic TF. Studying the clusters closely we are able to spot some potential outliers. Highlighting them using our outlier technique reveals that we, indeed, seem to have a number of data items that differ from the main cluster trends but, since the clusters are otherwise fairly homogeneous, we consider this an appropriate clustering.

Validating the clustering

We have now come to the conclusion that we are satisfied with our five clusters but we cannot be sure what might happen if another cluster were to be added. It may very well be the case that six clusters would look very different and be much more homogeneous. To test this, we simply add another cluster and the result is illustrated in Figure 10. Here, however, we see that the only major difference is that the cluster shown in Figure 9(c) (colour version in Figure 10(a)) splits into two clusters. These clusters are visualized with 2D parallel coordinates (Figure 10(b)) and 3D multi-relational parallel coordinates (Figure 10(c)). As could be expected, it is the cluster with most spread in a dimension that divides. Choosing the dimension with most spread as the focus dimension in 3D multi-relational parallel coordinates clearly illustrates the split in the focus dimension. The remaining outer dimensions are relatively compact and it is probably quite safe to conclude that six clusters are not appropriate.

Case study conclusions

As illustrated with this example, the use of high-precision textures together with different TFs is helpful when analysing larger multivariate data sets. Without using a TF, the analysis process would have been more tedious and might have led the user into making false judgements about the data. Also, since the TFs operate on high-precision textures, the feedback is instantaneous so the analysis can be interactively performed, regardless of the size of the data and its clusters. The combined view of 2D parallel coordinates and 3D multi-relational parallel coordinates further speeds up the analysis by enabling different views of the same feature. Since our work focuses on visualization methods, rather than on clustering algorithms, we have used a synthetic data set. Which clustering algorithm to choose when analysing a specific real world data set is beyond the scope of this paper.

Conclusions and future work

In this paper, our techniques for using high-precision textures and Transfer Functions (TFs) in conjunction with parallel coordinates (first described in [3]) have been extended to also work on unclustered data. Our techniques have also been incorporated into a 3D multi-relational parallel coordinates technique [4]. These two parallel coordinates techniques have then been integrated and can be efficiently used as a complement to each other during an analysis process. A case study has also been conducted and our preliminary results show that TFs can play an important role in the analysis of large multivariate data sets displayed using parallel coordinates. Non-linear and user-defined TFs can significantly help by revealing relevant structure information and would be extremely difficult to implement using direct rendering of lines.

We are currently investigating whether advanced image analysis techniques can be used on the high-precision textures to further aid in the analysis process. For future work we will study what additional interaction techniques could be beneficially incorporated into our current system.

Acknowledgements

This work has been funded by the Swedish Foundation for Strategic Research, grant A3 02:116.

References

[1] Inselberg A, Dimsdale B. Parallel coordinates: a tool for visualizing multi-dimensional geometry. *IEEE Visualization* 1990 (San Francisco, USA), IEEE Computer Society Press, 1990; 361–378.

[2] Inselberg A. The plane with parallel coordinates. *The Visual Computer* 1985; 1: 69–91.

[3] Johansson J, Ljung P, Jern M, Cooper M. Revealing structure within clustered parallel coordinates displays. *IEEE Symposium on Information Visualization* 2005 (Minneapolis, USA), IEEE Computer Society Press, 2005; 125–132.

[4] Johansson J, Cooper M, Jern M. 3-dimensional display for clustered multi-relational parallel coordinates. *9th International Conference on Information Visualization* 2005 (London, UK), IEEE Computer Society Press, 2005; 188–193.

[5] Hand D, Mannila H, Smyth P. *Principles of Data Mining*. MIT Press; 2001.

[6] Hastie T, Tibshirani R, Friedman JH. *The Elements of Statistical Learning*. Springer-Verlag; 2001.

[7] Hinterberger H. Data density: a powerful abstraction to manage and analyze multivariate data. *Ph.D. Thesis, Informatik-Dissertationen* ETH Zurich, No.4, 1987.

[8] Fua Y-H, Ward MO, Rundensteiner EA. Hierarchical parallel coordinates for exploration of large datasets. *IEEE Visualization* 1999 (San Francisco, USA), IEEE Computer Society Press, 1999; 43–50.

[9] Berthold MR, Hall LO. Visualizing fuzzy points in parallel coordinates. *IEEE Transactions on Fuzzy Systems* 2003; 11: 369–374.

[10] Andrienko G, Andrienko N. Parallel coordinates for exploring properties of subsets. *2nd International Conference on Coordinated and Multiple Views in Exploratory Visualization* 2004 (London, UK), IEEE Computer Society Press, 2004; 93–104.

[11] Novotny M. Visually effective information visualization of large data. *8th Central European Seminar on Computer Graphics* 2004 (Vienna, Austria), 41–48.

[12] Miller JJ, Wegman EJ. Construction of line densities for parallel coordinate plots. *Computing and Graphics in Statistics* 1992; 36: 107–123.

[13] Wegman EJ, Luo Q. High dimensional clustering using parallel coordinates and the grand tour. *Computing Science and Statistics* 28: 1997; 352–360.

[14] Rodrigues Jr. JF, Traina AJM, Traina Jr. C. Frequency plot and relevance plot to enhance visual data exploration. *XVI Brazilian Symposium on Computer Graphics and Image Processing* 2003, 117–124.

[15] Artero AO, de Oliveira MCF, Levkowitz H. Uncovering clusters in crowded parallel coordinates visualizations. *IEEE Symposium on Information Visualization* 2004 (Austin, USA), IEEE Computer Society Press, 2004; 81–88.

[16] Fanea E, Carpendale S, Isenberg T. An interactive 3D integration of parallel coordinates and star glyphs. *IEEE Symposium on Information Visualization* 2005 (Minneapolis, USA), IEEE Computer Society Press, 2005; 149–156.

[17] Gonzalez RC, Woods RE. *Digital Image Processing*. Prentice Hall; 2002.

[18] Pyle D. *Data Preparation for Data Mining*. Morgan Kaufmann; 1999.

[19] Han J, Kamber M. *Data Mining: Concepts and Techniques*. Morgan Kaufmann; 2001.

[20] Barlow N, Stuart LJ. Animator: a tool for the animation of parallel coordinates. *8th International Conference on Information Visualization* 2004 (London, UK), IEEE Computer Society Press, 2004; 725–730.

[21] Synthetic data set [WWW document] http://www.itn.liu.se/~jimjo/data/clusterdata.zip.

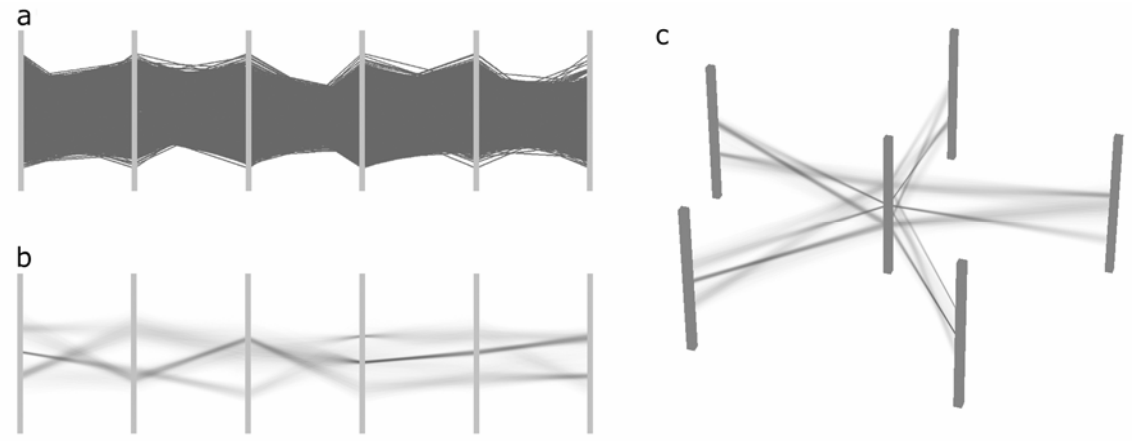

Figure 7: Visualizations of a synthetic data set. (a) When using standard parallel coordinates the cluttering is total and no structure can be seen. (b) A linear TF is used to visualize the unclustered data with standard parallel coordinates. (c) The same data set displayed using 3D multi-relational parallel coordinates.

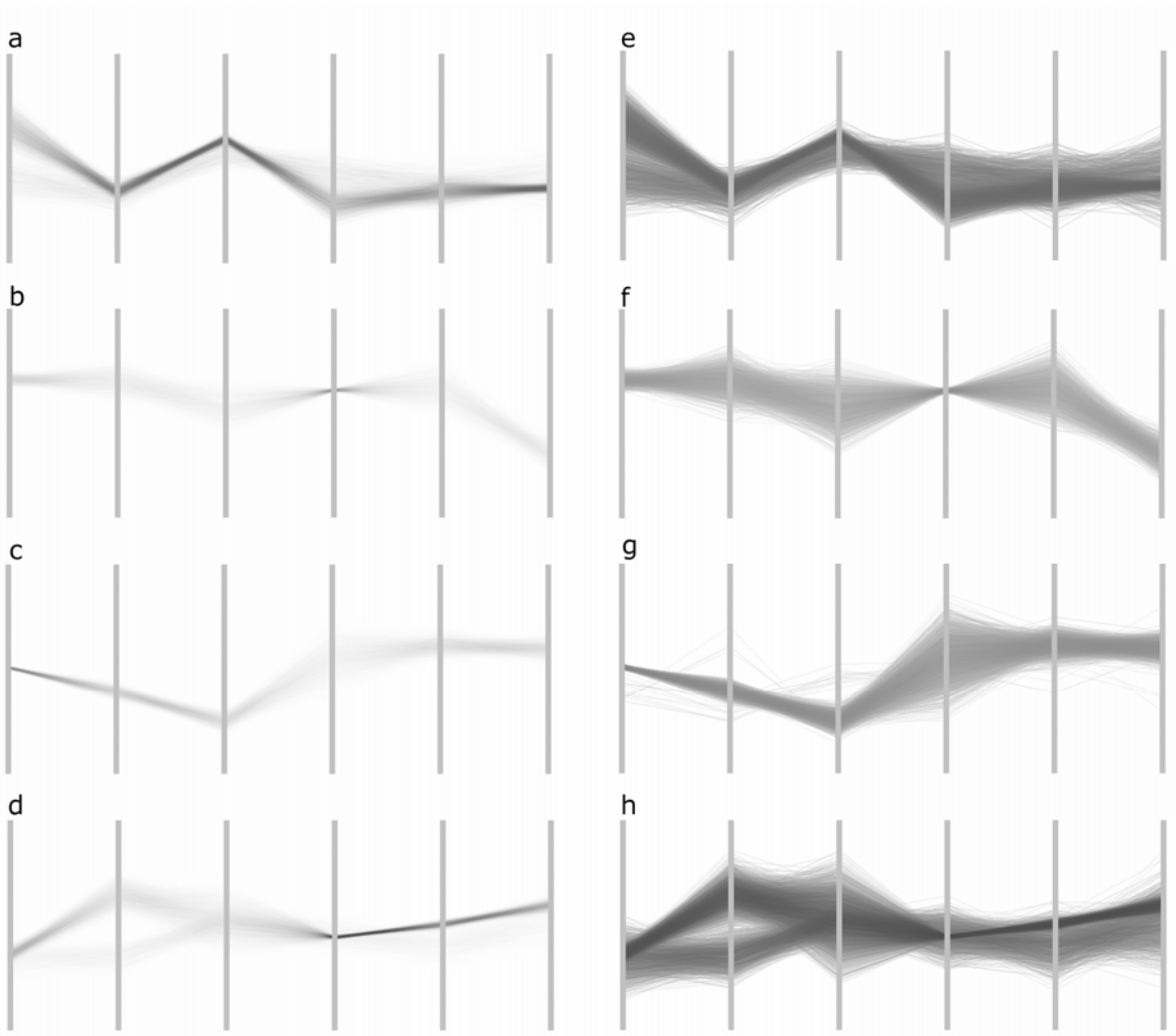

Figure 8: A close-up view of four clusters. Clusters in the left column (a–d) are visualized using a linear TF, while clusters in the right column (e–h) are visualized using a logarithmic TF which puts more emphasis on low-density regions.

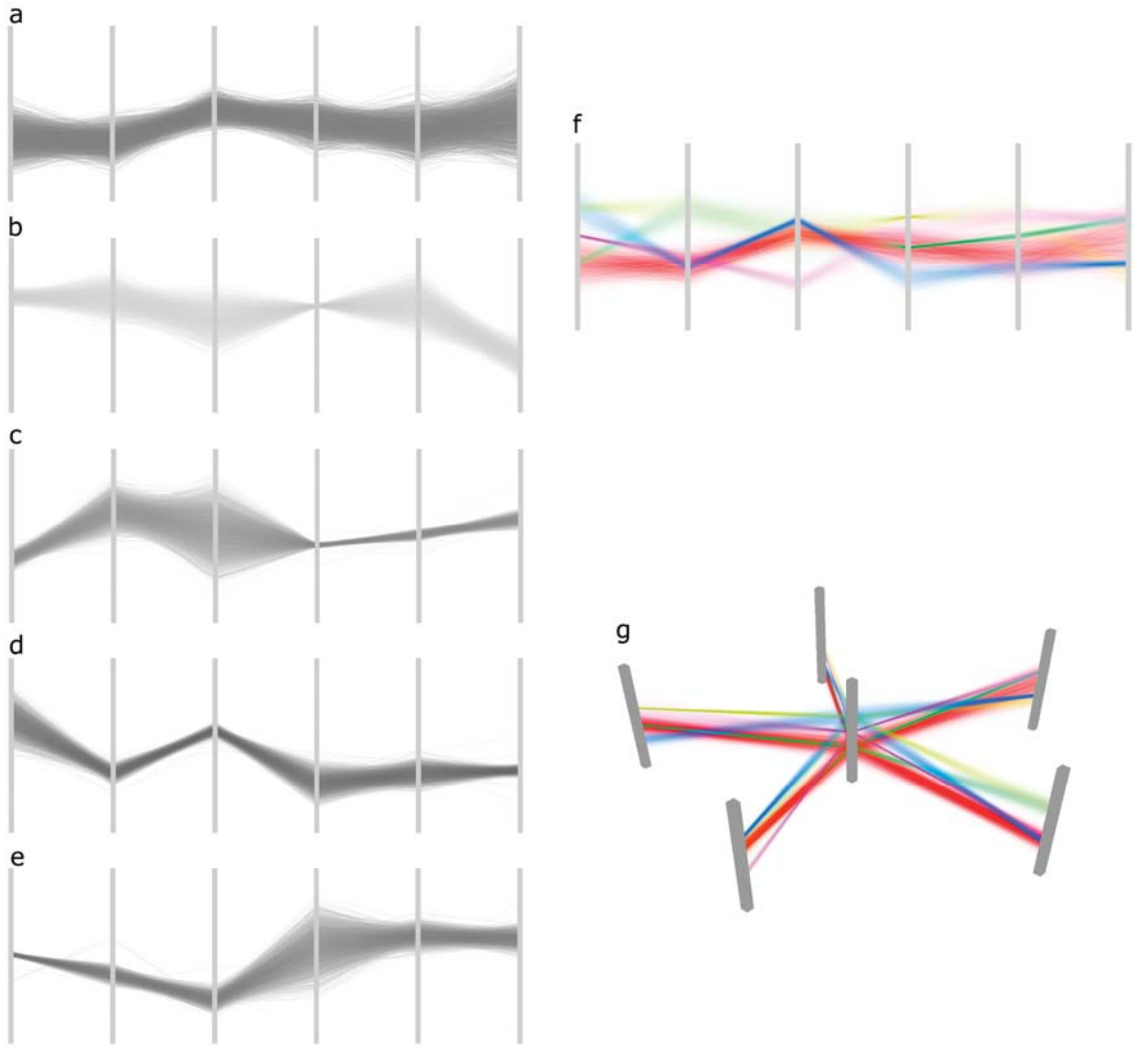

Figure 9: A close-up view of the five different clusters (a–e) visualized using a logarithmic TF. As can be seen, all clusters are now more homogeneous compared to the four clusters in Figure 8. (f) All clusters are simultaneously displayed in standard parallel coordinates using a linear TF. (g) The clusters are visualized in 3D multi-relational parallel coordinates.

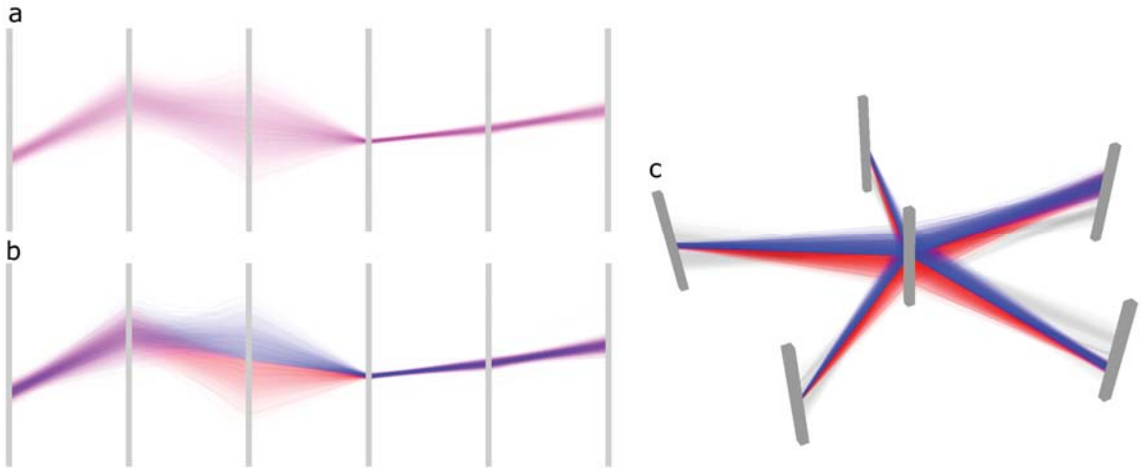

Figure 10: Requiring too many clusters causes the cluster with most spread to be split into two clusters. (a) Original cluster. (b) The cluster in (a) is split. (c) The 3D multi-relational parallel coordinates view of the split cluster. It is clear that these 2 clusters should be just one.