An Edge-Bundling Layout for Interactive Parallel Coordinates

Gregorio Palmas* MPI Informatik
Myroslav Bachynskyi† MPI Informatik Saarland University
Antti Oulasvirta‡ MPI Informatik Saarland University
Hans-Peter Seidel§ MPI Informatik
Tino Weinkauf¶ MPI Informatik

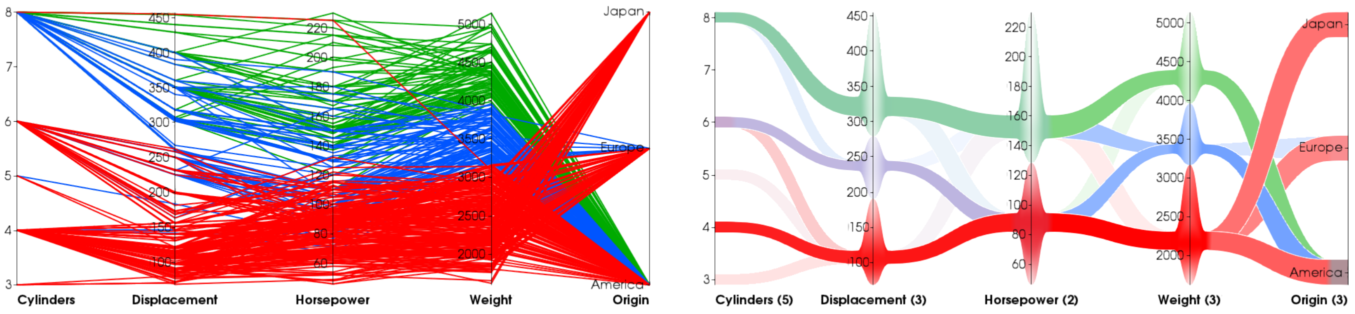


Figure 1: The *Cars* data set shown using the classic parallel coordinates plot (PCP) on the left, and our new edge-bundling layout on the right. Our method clusters the data in each dimension, and sets these clusters in relation to each other by bundling the lines between two axes. The bundles are then rendered using polygonal strips. This generates an abstract, clutter-reduced version of the classic PCP. We provide intuitive interactions such as the shown *axis-based selection*, where all clusters of a chosen axis are automatically selected using a different color. In this example, this has been done for the *Weight* axis. The colors will merge and split at other axes, thereby revealing the relations in a data set.

Abstract

Parallel Coordinates is an often used visualization method for multidimensional data sets. Its main challenges for large data sets are visual clutter and overplotting which hamper the recognition of patterns in the data. We present an edge-bundling method using density-based clustering for each dimension. This reduces clutter and provides a faster overview of clusters and trends. Moreover, it allows rendering the clustered lines using polygons, decreasing rendering time remarkably. In addition, we design interactions to support multidimensional clustering with this method. A user study shows improvements over the classic parallel coordinates plot in two user tasks: correlation estimation and subset tracing.

Index Terms: I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1 Introduction

The parallel coordinates system arranges several dimensions as parallel axes next to each other in the plane [10]. In contrast, the Cartesian coordinates system arranges its axes orthogonally. A parallel coordinates plot (PCP) [11, 23] of a multidimensional data set provides an overview of the relations between the dimensions. It has become a standard tool in visualization [6].

A classic PCP renders each observation point as a line. Especially for large data sets, this often creates visual clutter which makes it difficult to discern patterns in the data. A related issue is that lines are getting plotted on top of each other, which can even hide information.

An approach to mitigate these issues is data clustering. The aggregated information can then be used to render the PCP in an illustrative fashion using different forms of edge-bundling [7, 18, 25] or to distinguish between major trends and outliers [19]. This often leads to clutter-reduced and more informative visualizations.

However, these methods trade the more aggregated display of information with a loss of visual correspondence to the classic PCP. For example, previous edge-bundling methods bent the lines between two axes towards the center of a multidimensional cluster. But this cluster does not necessarily have an informative appearance between these two neighboring axes. In fact, it may not even reflect the relation between the two axes at all.

Our method clusters the data in each dimension independently. These clusters are shown on each axis and the lines are bundled to form polygonal strips between the clusters of neighboring axes. Hence, the clustering is directly related to the shown dimensions in every part of the plot. The result is an abstract, aggregated version of the classic PCP with less clutter. We expect this layout to be particularly helpful for tasks with large data sets that involve estimating correlations and identifying relationships across multiple dimensions.

Instead of rendering a line for each observation point, we render a bundle of lines as one polygonal strip. This makes the rendering time independent of the number of observation points. Hence, our method is responsive even for very large data sets.

Classic interactions with parallel coordinate plots include brushing of subsets, axis scaling, or axis reordering. They aid in revealing the complex relations in multidimensional data sets. We provide additional interactions with our method that make direct use of the per-axis clustering. Most notably, it is very fast with our method to brush subsets and form logical operations between them.

We conducted a user study comparing the classic PCP against our visualization. Users estimated the linear correlation between two dimensions, and they traced a subset across multiple dimensions. The results show statistically significant improvements in both tasks when using our visualization.

The paper is organized as follows: After reviewing related work in Section 2, we present our method in Section 3. The user study is discussed in Section 4. Finally, we discuss further results in Section 5, and draw conclusions in Section 6.

*e-mail: gpalmas@mpi-inf.mpg.de
†e-mail: mbachyns@mpi-inf.mpg.de
‡e-mail: oantti@mpi-inf.mpg.de
§e-mail: hpseidel@mpi-inf.mpg.de
¶e-mail: weinkauf@mpi-inf.mpg.de

2 Related Work

Since the thorough study of parallel coordinates by Inselberg [10] and its subsequent introduction as a visualization tool [11,23], it has become a standard for multidimensional data analysis and inspired a large amount of research. We concentrate here on the work that is most related to our approach and refer for more information to the state of the art report by Heinrich and Weiskopf [6].

Visual clutter due to overplotting is often cited as a major challenge for parallel coordinates [6]. Clustering the data is one possible approach to deal with it. Clusters can then be highlighted in the visualization using color or edge-bundling. The original edge-bundling method [8] uses cubic B-splines to show adjacency relations atop different tree visualization methods. This has been adapted to parallel coordinates plots by McDonnell and Mueller [18] such that the polylines become polycurves, which are bent between two axes towards the center of the cluster. This successfully frees up screen space and often creates clutter-reduced visualizations. The resulting curves are only C^0-continuous (not smooth) at an axis making it harder to visually trace them. This applies to the classic PCP as well. Heinrich et al. [7] solve this using a C^1-continuous bundling. Zhou et al. [25] present a bundling method that creates visual clusters without the need to cluster the data itself.

Existing work on edge-bundling concentrates on combining multidimensional clustering methods with parallel coordinates, but a multidimensional cluster does not necessarily have an informative appearance between two neighboring axes, nor does it necessarily reflect the relation between two axes at all. In contrast, we create an abstract version of the classic PCP, where we strive to retain its visual characteristics to some extent. Our method uses one-dimensional clusters, sets them in relation to each other between two axes, and enables the user to create higher-dimensional clusters using intuitive interactions.

Many clustering methods concentrate on the main trends and disregard outliers. But since outlier detection is of interest in some applications, Novotny and Hauser [19] consider the lines between two axes and distinguish them into clusters of similar lines and outliers. The clusters are then visualized using parallelograms and the outliers as lines. The method scales to very large data sets, since most observation points are rendered using a few parallelograms. Since the clustering is done in each segment separately, the parallelograms and their appearance may not be visually coherent at an axis, which makes tracing trends over several axes potentially difficult. In contrast, our method produces visually coherent results, since the one-dimensional clustering creates a setup around an axis, which is independent of the segment to the left or right.

In an effort to reduce visual clutter, a number of methods render polygons instead of lines. This goes back to Inselberg [10] who proposes to represent a multidimensional hypersurface via the envelope of its polylines in the PCP. The above mentioned [19] is another example. Fua et al. [4] render the PCP envelope of multidimensional clusters. McDonnell and Mueller [18] draw the envelopes of edge-bundled clusters in an illustrative fashion. Visual clutter can also be reduced by rendering into high dynamic range textures and applying different transfer functions to reveal otherwise hidden structures, as done by Johannsson et al. [13].

Kosara et al. [14] introduced the notion of *parallel sets*, where categorical data is rendered in a PCP using parallelograms. The thickness of the parallelogram reveals the number of observation points that are in both of the two connected categories. In our method, we follow [18] in using opacity to convey the number of observation points, since we concentrate on continuous data where the thickness of a connecting bundle or strip is strongly influenced by the metric in the parallel coordinates plot.

Several user studies assess the effectiveness of PCP versus scatter plots [9, 15, 16]. Kuang et al. [15] showed that PCP are better than the classic scatter plot in case of a low number of dimensions and a low density of the data set. Heinrich et al. [7] confirmed the effectiveness of edge-bundling versus the classic PCP. Johansson et al. [12] evaluated the capacity of humans to recognize patterns in PCP in the presence of different amounts of noise. Siirtola et al. [20] studied how first-time users acquired proficiency in using PCP. PCP have been evaluated for use in practical applications such as data base access [21] and interactive alarm filtering [1].

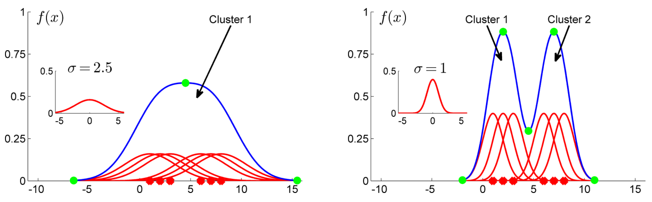


Figure 2: Simple example showing the Gaussian kernel density estimate for six observation points. A wide Gaussian kernel (left) leads to less clusters than a narrow kernel (right).

3 Method

Our method clusters the data for each variable (Section 3.1), bundles the polylines near each axis (Section 3.2), renders the bundles using polygonal strips (Section 3.3), and allows for several intuitive interactions (Section 3.4).

3.1 Data Clustering

We are clustering the observation points for each continuous variable independently. One such cluster represents observation points that are close to each other in that dimension. These points will later be bundled on the respective axis of the PCP.

We use a Gaussian kernel density estimation [22] to compute the density $f(\mathbf{x})$ of the observation points $\mathbf{x}_i$ for a given variable as follows:

$$f(\mathbf{x}) = \frac{1}{n\sigma\sqrt{2\pi}} \sum_{i=0}^{n} e^{-\frac{1}{2}\left(\frac{\mathbf{x}_i - \mathbf{x}}{\sigma}\right)^2}, \tag{1}$$

where σ refers to the standard deviation. A good intuition behind $f(\mathbf{x})$ is that it is high in areas with many observation points, and low in areas with few observation points.

Minima and maxima of $f(\mathbf{x})$ can be used to cluster the observation points. A maximum of $f(\mathbf{x})$ is a point of locally highest density. It is the center of a cluster. On the other hand, a minimum of $f(\mathbf{x})$ is a point of locally lowest density. It is the separation between two clusters. A cluster is then defined by the 3-tuple $(\mathbf{x}^-, \mathbf{x}^0, \mathbf{x}^+)$, where $\mathbf{x}^0$ is the location of the maximum, and $\mathbf{x}^-, \mathbf{x}^+$ are the locations of the two minima on either side.

The standard deviation σ can be seen as a scale-space parameter:

- A higher σ leads to a wider kernel, which leads to a wider impact of each observation point on the density $f(\mathbf{x})$. Consequently, $f(\mathbf{x})$ has less minima and maxima, and therefore also less clusters.

- A lower σ leads to a narrower kernel, which leads to a narrower impact of each observation point on the density $f(\mathbf{x})$. Consequently, $f(\mathbf{x})$ has more minima and maxima, and therefore also more clusters.

Note that this is a hierarchical clustering: we start at a single cluster for a high σ_1, i.e., $f(\mathbf{x})$ has one maximum and two minima to its left and right. Some $\sigma_2 < \sigma_1$ triggers a topological change of $f(\mathbf{x})$ by introducing a pair of one minimum and one maximum. This splits the previous cluster into two clusters. Figure 2 illustrates this. The

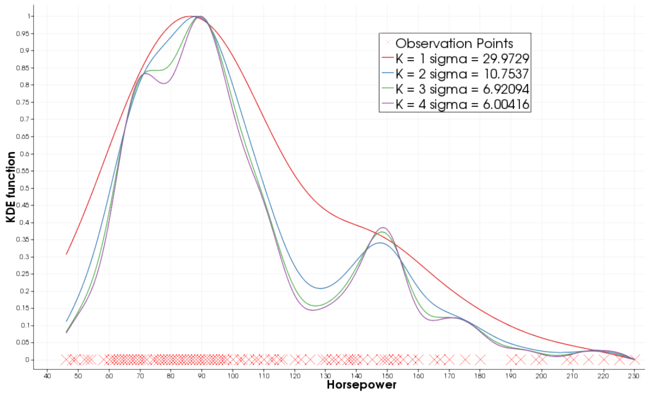


Figure 3: Normalized Gaussian kernel density estimations for different σ creating different numbers of clusters k for the *Horsepower* variable of the *Cars* data set.

process continues with $\sigma_3 < \sigma_2$, which splits one of the two clusters and leaves the other intact, i.e., we have three clusters for σ_3.

We use this in our user interface (see also Section 3.4): the user is able to choose the number of clusters k for each variable, thereby effectively varying the level of detail used to represent a variable.

For each variable, we precompute the clusterings for $k = 1,\ldots,k_n$ number of clusters. To do so, we densely sample σ to find the σ_k at which a cluster splits (let k denote the number of clusters for a given σ_k). This is done until we computed $\sigma_1,\ldots,\sigma_{k_n}$. For most data sets we found it sufficient to sample σ 2000 times between 25% and 1% of the data range.

The precomputation is the most expensive part of our method, but still feasible. One variable with 10^5 observation points needs less than a minute on current desktop hardware in a single computing thread (for $k_n = 99$). We use OpenMP to precompute the clustering for several variables in parallel. For the well-known *Cars* data set [24] (5 continuous variables, ≈400 observation points), the precomputation is done in half a second. Figure 3 shows the density functions for $\sigma_1,\ldots,\sigma_4$ for a variable of this data set.

Categorical variables are not clustered using the above method. Instead, we treat each category as a cluster.

We could also use any other clustering method, since the visual layout of the plot is independent of that (see next section). We chose this particular method for two reasons: (1) its hierarchical nature is intuitive, since the clusters for k and $k+1$ are nested. This makes the effects of increasing/decreasing the number of clusters predictable to the user. And (2), the density estimate will be used later for rendering (Section 3.3).

3.2 Visual Layout using Edge-Bundling

In a classic parallel coordinates plot, the observation points are drawn as polylines. Our new layout treats them as cubic Bézier splines. The purpose is to bundle them according to the clustering in the vicinity of each axis, and to provide smooth curves that are easy to follow with the eyes. In this section, we describe our edge-bundling layout by means of actually drawing a curve for each observation point. In the next section, we will see how this can be simplified to drawing a small number of polygonal strips.

Consider two neighboring axes A and B of a parallel coordinates plot and the area between them. We place a virtual bundling axis nearby each data axis, as illustrated in Figure 4. Let us denote them as A' and B'. Note how this segments the area between the data axes into three different parts. The distance between a data axis and its bundling axis is a parameter in our software, but we keep it fixed to 10 percent of the distance between the data axes for all screenshots in this paper.

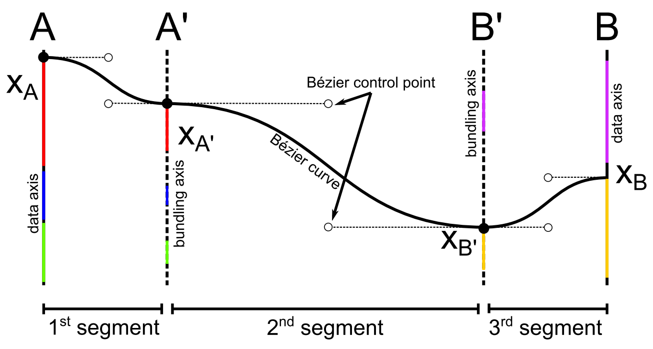
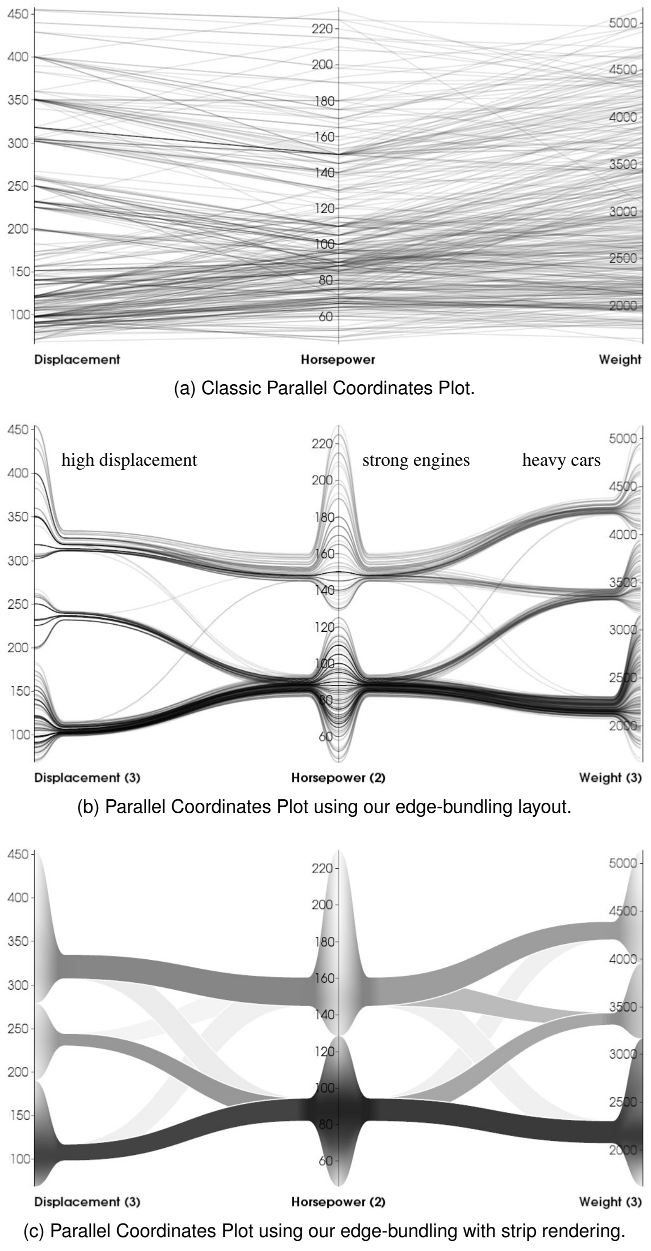


Figure 4: Visual layout between two data axes A and B. Edges are bundled near each data axis according to the clusters in each dimension. To do so, we form a cubic Bézier spline out of three segments.

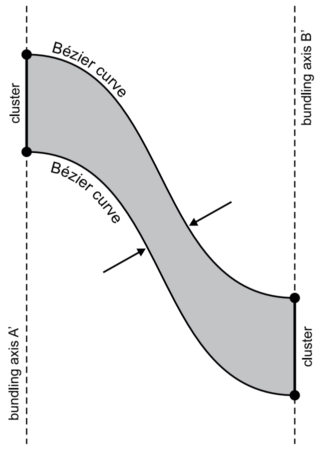


(a) Classic Parallel Coordinates Plot.

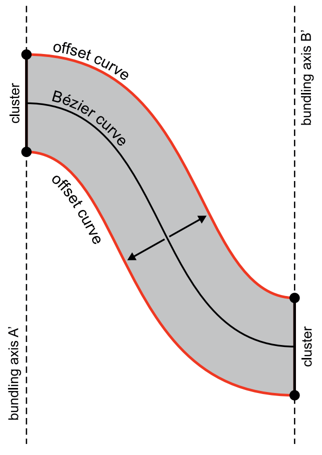
(b) Parallel Coordinates Plot using our edge-bundling layout.

(c) Parallel Coordinates Plot using our edge-bundling with strip rendering.

Figure 5: Comparison of the different PCP layouts. Shown are three dimensions from the *Cars* data set.



(a) Two bent Bézier curves do not keep a constant distance from each other.



(b) Polygonal strip with constant width between two clusters constructed by offsetting a Bézier curve.

Figure 6: Construction of polygonal strips between clusters, which eliminate the need to draw a Bézier curve for each observation point.

Recall from the previous section that a cluster is represented on the data axis by the 3-tuple $(\mathbf{x}^-, \mathbf{x}^0, \mathbf{x}^+)$. On the bundling axis, we represent a cluster as well, but with a smaller segment:

$$\left(\mathbf{x}^0 + s(\mathbf{x}^- - \mathbf{x}^0)\,, \quad \mathbf{x}^0\,, \quad \mathbf{x}^0 + s(\mathbf{x}^+ - \mathbf{x}^0)\right) \tag{2}$$

where s is a scaling factor, which we keep constant throughout the paper at $s = 0.15$. Clusters are shown in Figure 4 as colored segments on an axis. A point $\mathbf{x}_A$ on the data axis is represented in its cluster on the bundling axis at $\mathbf{x}_{A'} = \mathbf{x}^0 + s(\mathbf{x}_A - \mathbf{x}^0)$.

Consider an observation point $(\mathbf{x}_A,\ \mathbf{x}_B)$. The classic PCP represents it using a straight line. In our layout, we create three cubic Bézier spline segments (Figure 4). The first segment starts from the data axis A at $\mathbf{x}_A$ and ends at the bundling axis A' at the point $\mathbf{x}_{A'}$. The second segment starts at $\mathbf{x}_{A'}$ and ends at $\mathbf{x}_{B'}$. The third segment runs from $\mathbf{x}_{B'}$ to $\mathbf{x}_B$.

The two additional Bézier points per cubic segment are defined such that we obtain a G^1-continuous spline. As illustrated in Figure 4, this can easily be done by horizontally offsetting the start and end point of each segment. Note how this setup creates Bézier splines that are not only smooth between two data axes A and B, but also across a data axis. This way, the curve of an observation point is easy to follow with the eyes.

Figures 5a–b visualize the *Cars* data set using the classic PCP and the PCP with our edge-bundling layout. Loosely spoken, we bundle the curves near a data axis, then we bring these bundles to the next data axis, where we un-bundle them. Note how this immediately reveals the relationships in this data set: engines with a high displacement create lots of horsepower, which is in turn needed to drive heavy cars. The bundled lines are easy to follow and they elucidate the relationships between the clusters on each axis.

3.3 Representation using Polygonal Strips

It is easy to see that our edge-bundling layout has a smaller visual footprint than the classic PCP, i.e., less pixels are filled. This leads to less visual clutter for many data sets. However, drawing a curve for each observation point has three disadvantages:

- The bundling brings the curves closer to each other, which actually increases the amount of overplotting within a bundle.
- The rendering performance depends on the number of lines. This is also true for the classic PCP. Hence, data sets with a very large number of observation points cannot be explored interactively.
- Two Bézier curves starting and ending with the same vertical offset do not have the same distance from each other over the course of their run: the more they are bent, the closer they come to each other in the middle (Figure 6a). In other words, a highly bent line bundle may become very thin.

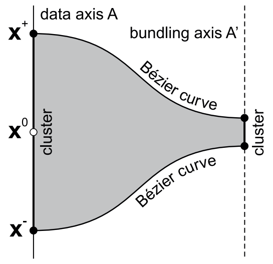


Figure 7: Construction of the polygonal strip between a data axis and its bundling axis. This defines the appearance of the clusters around each axis.

We solve these issues by rendering a small set of polygonal strips instead of the large number of Bézier curves. This is based on the observation that our edge-bundling layout creates a maximum of $k_A \times k_B$ bundles between two axes, where k_A, k_B refer to the number of clusters on either axis. Hence, we draw a strip between two clusters if there is at least one observation point in both clusters.

The polygonal strip is created using two offset curves as illustrated in Figure 6b. First, we compute a Bézier curve between the centers of the two clusters on either bundling axis. For each sample on the Bézier curve, we create a point on an offset curve by moving perpendicular to the tangent of the Bézier curve. The offset distance in Figure 6b is constant, since both clusters have the same size. If they do not have the same size, then we linearly interpolate between them to get the offset distance along the strip. Note that these offset curves guarantee that the width of the strip depends only on the size of the connected clusters, but not on how much it is bent. Furthermore, note that the offset curves cannot be represented as Bézier curves, in general.

It remains to draw the actual cluster between the data axis and its bundling axis. We use the setup shown in Figure 7 where the outermost Bézier curves define the strip.

We triangulate the strips and render them using OpenGL. This provides fast performance and hardware antialiasing. Furthermore, we make use of the following graphical attributes:

- The grayscale value of the rendered strip reflects the number of observation points in that strip.
- The strips can only intersect between two bundling axes. We render them in the order of increasing number of observation points.
- We render halos around the strips for a better disambiguation at intersections.

Figure 5c shows an example from the *Cars* data set.

Note how the issues from the beginning of this section are addressed by our solution:

- There is no overplotting of lines anymore. Of course, the problem has been shifted to the grayscale values of the strips (number of observation points), which may have a large variation, i.e., a high dynamic range. We provide simple tools such as gamma correction in our software to mitigate this issue. Our experience shows that it is easier to deal with that than with overplotting lines.

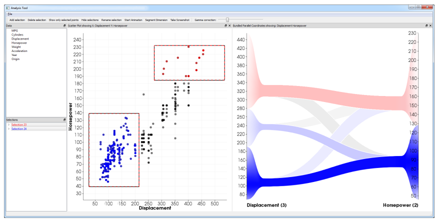
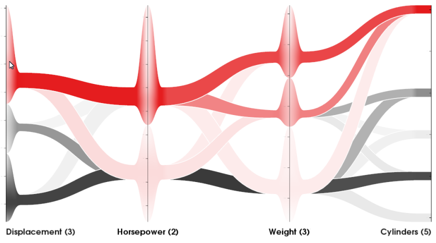
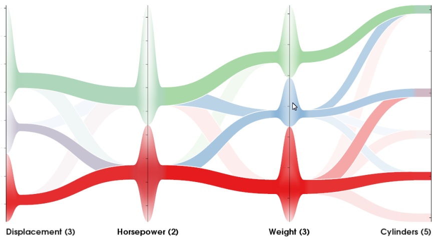


Figure 8: Arbitrary selections specified in other visualizations are shown coherently by our polygonal strips using color blending. This way, our method can be applied in *Linking & Brushing* scenarios.



(a) Hovering over a cluster creates a selection of all its observation points. This quickly reveals their distribution on other axes.



(b) Hovering over an axis automatically selects all its clusters with a different color. Here, we hover with the mouse cursor over the weight axis. Note how the colors merge in other regions, e.g., the violet color at the left.

Figure 9: Interactive quick selection modes. Hovering with the cursor over a cluster or axis automatically highlights parts of the data set.

- The rendering performance depends on the number of strips, which in turn depends on the number of clusters. Even data sets with a very large number of observation points can be explored interactively.

- The strips between the bundling axes have a well-defined width due to the offset curves.

In a classic PCP, a selection of observation points can easily be highlighted by drawing the corresponding lines in a different color. We can also highlight selections with our setup. To do so, we need a representation of the observation points at each cluster: a bitset records for each observation point whether it is in this cluster or not. A selection of observation points can also be given as a bitset. A simple logical AND combination between the cluster bitset and the selection bitset gives a new bitset with the selected points in that cluster. This result is further used to obtain a bitset describing the selected points in the strip between two clusters (again, a simple AND combination). Finally, we colorize the strips according to the fraction of selected / total number of observation points in that strip. In case of more than one selection, we blend the colors accordingly.

Hence, our method can be used in *Linking & Brushing* scenarios [2] to show arbitrary selections specified in other visualization techniques such as scatter plots (Figure 8). Since we use boost's bitset implementation, we have a rather small memory overhead (1 bit per observation point and cluster) and very fast logic operations. We found this setup to be very responsive, even for very large data sets. Please see the supplemental material for screen capture videos showing this.

Our current implementation does not support smooth selections [3, 5, 17], but it is principally possible at the expense of doing the slower fuzzy logic.

3.4 User Interactions

Our method allows for the typical interactions of classic PCP: scaling an axis, changing the order of the axes, or brushing (selecting) the observation points in a range of a variable.

Due to the clustering and the rendering of strips, we can provide some additional interactions:

- A quick selection mode is available, where hovering with the mouse over any cluster or bundle strip selects the respective observation points. They are then highlighted in the entire plot as discussed in the previous section. Figure 9a shows this.

- An axis-based selection mode is available, where hovering over an axis creates several selections at once: each cluster on that axis is automatically selected with a different color. Figure 9b shows this. Note how the colors blend at locations where the observation points from two different clusters merge.

- It is also straightforward to create logic combinations of clusters. To do so, one simply clicks on the clusters in the desired order. Holding down the Shift- or Ctrl-key determines the type of the logic operation (AND, OR).

- The user can interactively change the number of clusters per variable. In our implementation, this is done using the mouse wheel.

Please see the videos in the supplemental material for an impression of these interactions. Note that the above interactions limit the definition of selections to clusters, bundle strips, and their combinations. To define arbitrary selections, we use the typical brushing on an axis known from classic PCP.

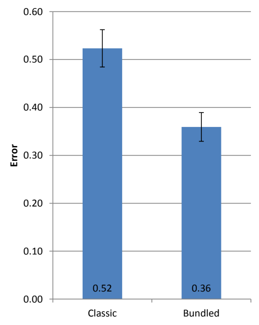
4 User Study

We conducted a comparative user study to assess the user performance in two visualization tasks that are typical to PCP. The study was done using static images shown on a web questionnaire. For comparison, each respondent did each task with both our method and the classic PCP. Two tasks were used:

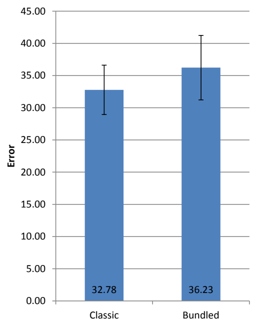
- **Task 1: Correlation estimation**
  This task asks users to identify the linear correlation of two variables, which is a value that varies continuously in the range $[-1,+1]$ as follows:
  - $+1$ denotes a positive correlation. Example: Earning *more* money leads to *more* money in the bank.
  - 0 denotes that there is no correlation. Example: Earning more money has no effect on the outside temparature.
  - -1 denotes a negative correlation. Example: Spending *more* money leads to *less* money in the bank.

- **Task 2: Subset tracing**
  This task asks users to follow some observation points over several axes in the plot. This task was done in two conditions: with and without color for selection. While we did not test user interactions directly, highlighting a selection using color shows the *effect* of an user interaction.

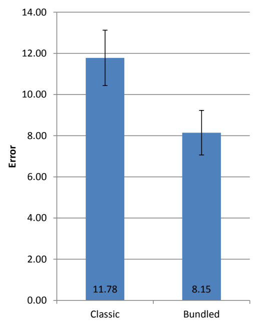
  For an example of this task, consider a data set that tells us where each *student* and each *professor* have lunch at the university: in the canteen, in a restaurant, or in the office. We asked the participants to estimate the percentage of *students* that eat in the canteen, in a restaurant, or in the office. To do so, they had to follow the lines or polygons in either visualization.



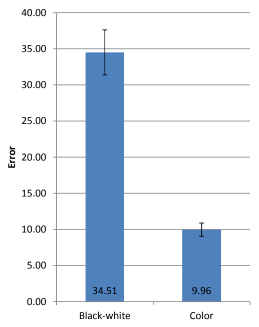
(a) Task 1: Judging a correlation between two axes.



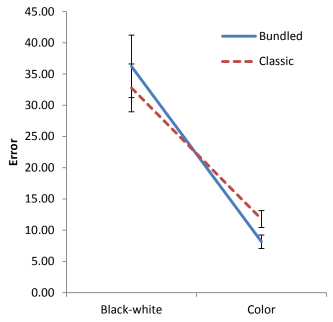
(b) Task 2: Tracing a subset, without color.



(c) Task 2: Tracing a subset, with color.



(d) Task 2: Main effect of color presence.



(e) Task 2: Interactions between the two independent variables.

Figure 10: The results of our user study show that our bundled PCP increases the users' accuracy in complex visual data analysis tasks. The plots show error measures: lower values are better. The vertical bars denote the 95% confidence interval. See the text for details.

4.1 Method

The link to the questionnaire was sent to computer science students and researchers at a local university. Task 1 has 82 valid respondents, Task 2 has 55.

Task 1 has one independent variable, *Visualization*, with two levels: classic PCP vs. bundled PCP. Task 2 has two independent variables: *Visualization* (2 levels) and *Color* (color/black-white).

The *Cars* data set and some synthetic examples (e.g., with a fully positive/negative correlation) have been used. In Task 1, the participants had to judge the correlation of two variables and make a decision on a discrete scale $(-1, -0.75, -0.5, -0.25, 0, 0.25, 0.5, 0.75, 1)$. In Task 2, we asked the participants to trace a subset of the data over several dimensions. Start and end axes were categorical. The subset to follow was defined as a category. There were two metric axes between the left and right categorical axes. The right categorical axis had always three categories, i.e., we asked for three numbers summing up to 100%.

Every task was carefully explained to the participants. Furthermore, the actual task was preceded by a simple toy example that was used as a screening criterion. An incorrect answer implies that a participant has either a poor understanding of parallel coordinates, or is not motivated to complete the task. To avoid learning effects, a specific data arrangement was never shown twice. The order in which the methods were presented to the user was randomized for each task.

4.2 Results

To preprocess the data, we removed the data of respondents who completed the tasks only partially.

For the remaining respondent pool, we calculated the *mean error* in each task (see below) and in each condition. We then screened out participants with a mean error larger than the mean +2 standard deviations of the whole population.

The dependent variables were computed for the two tasks as follows:

- Task 1: As error measure, we calculated the absolute difference between the response and the ground-truth correlation (Pearson correlation coefficient). 95 participants have completed Task 1. 8 of them were outliers, and 5 failed the screening questions with a mean error larger than 0.5.

- Task 2: The error measure was calculated as Euclidean distance between the responses and the ground-truth percentages. 61 participants completed this task, out of which 6 were outliers.

4.2.1 Task 1: Correlation estimation

To test the effect of *Visualization*, we performed a paired t-test. The error was by 0.16 ± 0.04 (31%) smaller for our bundled PCP ($\mu = 0.36$) than for the classic PCP ($\mu = 0.52$). This difference was statistically significant: $t(81) = 7.57$, $p < 0.001$. See Figure 10a.

4.2.2 Task 2: Subset tracing

Figures 10b–c show the mean and the standard deviation for this task for all four conditions.

To test the effects of *Visualization* and *Color*, we carried out a two-way ANOVA with the two independent variables. There was no main effect of *Visualization*: $F(1,54) = 0.0035$, $p = 0.95$. However, the presence of color decreased the average error by 24.54 ± 3.33 (71%) from 34.51 to 9.96, as shown in Figure 10d. The main effect of *Color* was statistically significant: $F(1,54) = 215.36$, $p < 0.001$.

Interestingly, there was a significant interaction effect between *Color* and *Visualization*: $F(1,54) = 4.68$, $p = 0.035$. As it can be seen in Figure 10e, our bundled layout benefits more from the presence of color than the classic PCP. User's error decreases 78% with our technique and only 64% with the classic PCP.

4.3 Interpretation

To sum up, the bundled PCP increases the users' accuracy in complex visual data analysis tasks. The users had a statistically significant lower error for judging the correlation between two variables, and for tracing a subset when color was present. When color was not available for tracing a subset, the performance was at the same level as with the classic PCP. The use of color is particularly useful for tracing in the bundled PCP.

5 Further Results and Discussion

Figure 11 shows the *out5d* data set [24], which contains remote sensing data from a region in western Australia. The data set contains 16384 observation points and is an example for the overplotting issue of the classic PCP. The top row of Figure 11 shows the classic PCP and our method without a selection, i.e., as grayscale visualizations. Here, the overplotting in the classic PCP hides important relations. An example is the relation between thorium and uranium shown as the two right-most axes. The classic PCP is almost entirely black between these axes, and makes it therefore impossible to draw conclusions about the data. Our method, on the other hand, clearly shows a positive correlation between thorium and uranium. This can be attributed to the explicit aggregation when converting to the strip-based representation.

Using color to highlight single selections reveals more information for both methods – as we already found out in our user study

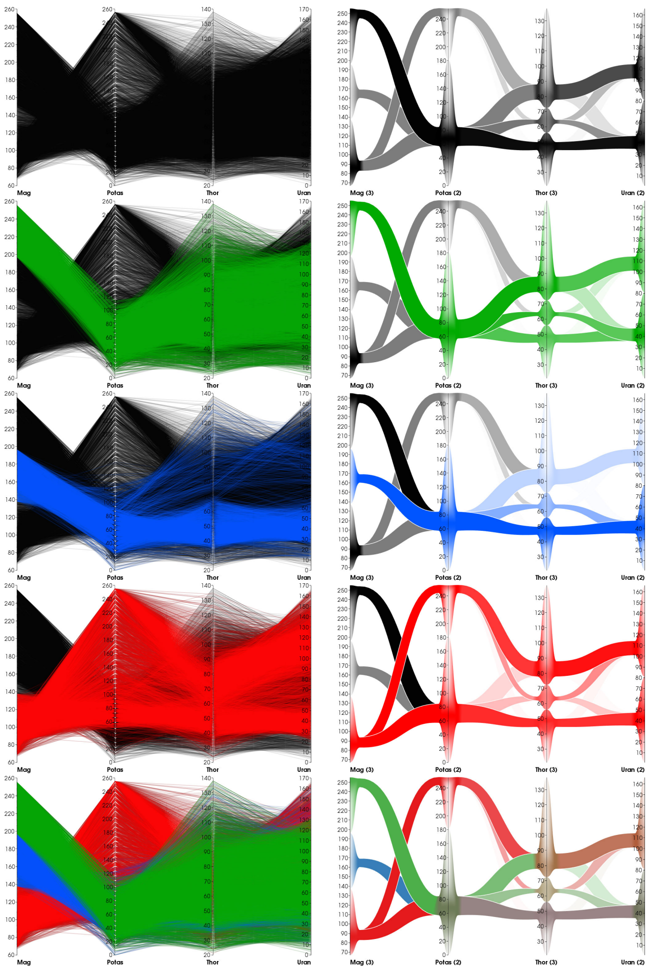
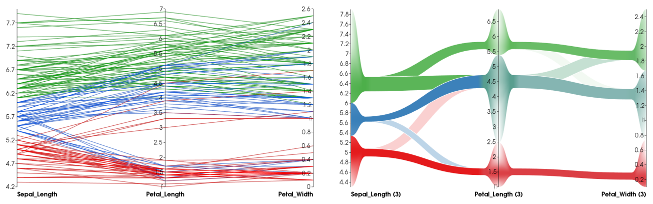


Figure 11: The *out5d* data set with 16384 observation points shown without selection, with single selections, and with several selections at once. Left: classic PCP. Right: our method.

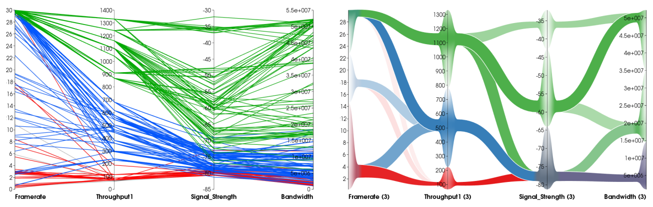
–, but showing all three selections at once (last row of Figure 11) works better for our method due to the explicit merging of colors instead of the simple overplotting in the classic version. In this example, the three clusters on the left-most axis have been assigned the primary colors.

The *Iris* and the *netperf* data sets [24] are shown in Figure 12. They are quite small data sets with 150 and 179 observation points, respectively. They show that our method creates an abstract version of the classic PCP – thereby retaining some of its visual characteristics. For example, note the large empty space between the two right-most axes of Figure 12a: both methods show it in a similar way. Note the dominance of blue and green in Figure 12b and how the shape of these selections is similar for both methods. Similar observations can be made for the *Cars* data set in Figure 5.

Comparison to other edge-bundling approach Figure 13 shows the *netperf* data set using the edge-bundling method of McDonnell and Mueller [18]. In contrast to our method, the bundling is based on a clustering defined for all dimensions at once. Hence, this method is well-suited for showing and distinguishing multidimensional clusters atop a PCP. In general, these clusters intersect each other in the PCP visualization space and project to non-contiguous subsets on each axis. This is contrary to our approach, which works with contiguous clusters on each axis. This enables us to bundle the lines *around* each axis, whereas [18] bundles *between* two axes.



(a) *Iris* data set.



(b) *Netperf* data set.

Figure 12: Our method is an abstract version of the classic PCP. It retains some of its visual characteristics and adds new possibilities such as faster interactions.

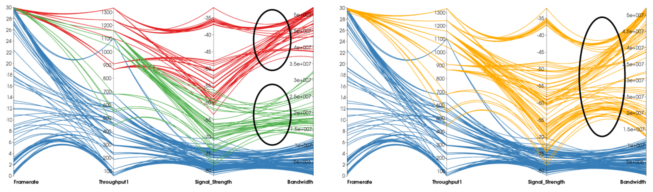


Figure 13: The *netperf* data set visualized using the edge-bundling method of McDonnell and Mueller [18]. Left: k-means clustering for $k = 3$. Note how the multidimensional clusters overlap on several axes. Right: When merging two clusters (red+green=orange), the edge-bundling of [18] changes the bending of the curves.

Both methods deal with large data by rendering sets of lines as polygons. For [18], the non-contiguous subsets on each axis manifest as holes within their polygons (see Figures 8, 9, 10, 11 in [18]).

Our method lends itself to direct interactions with the clusters, since the polygonal strips are easy-to-click targets. Creating a multidimensional cluster using the described interactions (Section 3.4) is a matter of a few clicks. Interactions with clusters have not been described in [18], but it is easy to see that merging two clusters (OR combination) leads to a different bending of each curve in their setup. In other words, such interactions would lead to a visually incoherent behavior that is difficult to predict by the user.

In summary, while [18] also uses clustering and bundling technologies, their approach is orthogonal to ours. In fact, it is an interesting question for future research how multidimensional clusters can be visualized on top of our method by bending our strips similar to how it is done in [18].

Large data sets Figure 14 shows a rather large data set with more than 100000 observation points. It contains biomechanical simulation data. Interacting with the classic PCP becomes painstakingly slow for such large data sets. Note that this is not an implementation or hardware issue, but a fundamental matter: the rendering speed depends on the number of observation points. In contrast, our method is responsive, because it only depends on the number of clusters. In the example from Figure 14, a trained user needed 15 seconds to create a multidimensional selection using our method, versus 40 seconds to do the same with the classic PCP. See also the supplemental video for a demonstration. Also note that the actual selection action in our method can be a simple mouse click on a

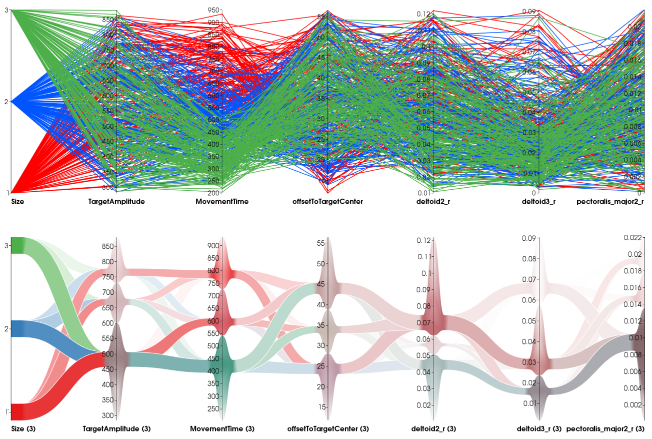


Figure 14: Biomechanical data set with 116732 observation points.

cluster (or a strip between two clusters). Contrary, the classic PCP requires a dragging motion for a selection.

6 Conclusion and Future Work

In this paper, we gave the following contributions:

- An edge-bundling method for parallel coordinates plots, which is built on one-dimensional clustering.
- A set of interactions for this method.
- A user study comparing our method to the classic PCP, and evaluating the utility of color in PCP plots.

The goal of our work was to address well-known shortcomings of PCP: visual clutter and overplotting. We addressed these with an edge-bundling method that utilizes one-dimensional clustering on each axis. This way, our edge-bundling leads to an abstract version of the classic PCP. The bundles are rendered as polygonal strips, which leads to a clutter-reduced visualization and the rendering speed becomes independent of the number of observation points. We have shown how the clusters on each axis and the bundle strips between the axes are useful as handles for intuitive and fast interactions. Multidimensional clusters can be created and trends isolated with simple gestures.

The examples and the data from the user study show that the method is effectively increasing the analysis performance of users. In particular, we found empirical evidence for improvements in user performance in correlation judgment tasks and subset tracing tasks.

An obvious limitation of our method is that it works best for a small or medium amount of clusters per variable, say 2-5 clusters per axis. More clusters create too much visual clutter. If there is just one cluster, then bundling the edges around that axis is useless. In this sense, it is important to choose a clustering method that works well with a specific data set. One may actually choose different clustering methods for different variables within the same data set, since our subsequent visualization is not affected by that. However, clustering data is always an application-dependent issue.

One of the virtues of our method is its aggregation capability: the big trends are highlighted. By design, this introduces a certain level of abstraction where individual observation points cannot be seen anymore. In particular, outliers are not taken explicitly care of at the moment. It would be interesting for future work to investigate how the principal idea behind the outlier preservation of Novotny and Hauser [19] can be applied to our method as well.

Acknowledgements

We thank Himangshu Saikia for his help with the user study.

References

[1] S. Azhar and M. Rissanen. Evaluation of parallel coordinates for interactive alarm filtering. In *Information Visualisation (IV), 2011 15th International Conference on*, pages 102–109, 2011.

[2] R. Becker and W. Cleveland. Brushing scatterplots. *Technometrics*, 29(2):127–142, 1987.

[3] H. Doleisch, M. Gasser, and H. Hauser. Interactive feature specification for focus+context visualization of complex simulation data. In *Proc. VisSym 03*, pages 239–248, 2003.

[4] Y.-H. Fua, M. O. Ward, and E. A. Rundensteiner. Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of the conference on Visualization '99: celebrating ten years*, VIS '99, pages 43–50, Los Alamitos, CA, USA, 1999. IEEE Computer Society Press.

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

[6] D. Heinrich and D. Weiskopf. State of the art of parallel coordinates. In *Proc. Eurographics*, pages 95–116, 2013.

[7] J. Heinrich, Y. Luo, A. E. Kirkpatrick, H. Zhang, and D. Weiskopf. Evaluation of a bundling technique for parallel coordinates. *CoRR*, abs/1109.6073, 2011.

[8] D. Holten. Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):741–748, 2006.

[9] D. Holten and J. J. Van Wijk. Evaluation of cluster identification performance for different pcp variants. *Computer Graphics Forum*, 29(3):793–802, 2010.

[10] A. Inselberg. The plane with parallel coordinates. *The Visual Computer*, 1(4):69–91, 1985.

[11] A. Inselberg and B. Dimsdale. Parallel coordinates: A tool for visualizing multi-dimensional geometry. In *Proc. IEEE Visualization '90*, pages 361–375, Los Alamitos, CA, USA, 1990. IEEE Computer Society Press.

[12] J. Johansson, C. Forsell, M. Lind, and M. Cooper. Perceiving patterns in parallel coordinates: Determining thresholds for identification of relationships. *Information Visualization*, 7(2):152–162, 2008.

[13] J. Johansson, P. Ljung, M. Jern, and M. Cooper. Revealing structure within clustered parallel coordinates displays. In *Proc. Information Visualization*, pages 125–132, 2005.

[14] R. Kosara, F. Bendix, and H. Hauser. Parallel sets: interactive exploration and visual analysis of categorical data. *Visualization and Computer Graphics, IEEE Transactions on*, 12(4):558–568, 2006.

[15] X. Kuang, H. Zhang, S. Zhao, and M. McGuffin. Tracing tuples across dimensions: A comparison of scatterplots and parallel coordinate plots. *Computer Graphics Forum*, 31(3pt4):1365–1374, 2012.

[16] J. Li, J.-B. Martens, and J. J. van Wijk. Judging correlation from scatterplots and parallel coordinate plots. *Information Visualization*, 9(1):13–30, 2010.

[17] A. R. Martin and M. O. Ward. High dimensional brushing for interactive exploration of multivariate data. In *Proc. IEEE Visualization*, pages 271–278, 1995.

[18] K. T. McDonnell and K. Mueller. Illustrative parallel coordinates. *Computer Graphics Forum*, 27(3):1031–1038, 2008.

[19] M. Novotny and H. Hauser. Outlier-preserving focus+context visualization in parallel coordinates. *Visualization and Computer Graphics, IEEE Transactions on*, 12(5):893–900, 2006.

[20] H. Siirtola, T. Laivo, T. Heimonen, and K.-J. Räihä. Visual perception of parallel coordinate visualizations. In *Information Visualisation, 2009 13th International Conference*, pages 3–9, 2009.

[21] H. Siirtola and K.-J. Räihä. Interacting with parallel coordinates. *Interacting with Computers*, 18(6):1278–1309, 2006.

[22] B. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman & Hall/CRC, 1986.

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

[24] XmdvTool homepage, http://davis.wpi.edu/xmdv/datasets.html.

[25] H. Zhou, X. Yuan, H. Qu, W. Cui, and B. Chen. Visual clustering in parallel coordinates. *Computer Graphics Forum*, 27(3):1047–1054, 2008.