Outlier-preserving Focus+Context Visualization in Parallel Coordinates

Matej Novotný and Helwig Hauser

Abstract—Focus+context visualization integrates a visually accentuated representation of selected data items *in focus* (more details, more opacity, etc.) with a visually deemphasized representation of the rest of the data, i.e., the *context*. The role of context visualization is to provide an overview of the data for improved user orientation and improved navigation. A good overview comprises the representation of both *outliers* and *trends*. Up to now, however, context visualization not really treated outliers sufficiently. In this paper we present a new approach to focus+context visualization in parallel coordinates which is truthful to outliers in the sense that small-scale features are detected before visualization and then treated specially during context visualization. Generally, we present a solution which enables context visualization at several levels of abstraction, both for the representation of outliers and trends. We introduce outlier detection and context generation to parallel coordinates on the basis of a binned data representation. This leads to an output-oriented visualization approach which means that only those parts of the visualization process are executed which actually affect the final rendering. Accordingly, the performance of this solution is much more dependent on the visualization size than on the data size which makes it especially interesting for large datasets. Previous approaches are outperformed, the new solution was successfully applied to datasets with up to 3 million data records and up to 50 dimensions.

Index Terms—Parallel coordinates, focus+context visualization, outliers & trends, large data visualization.

1 Introduction

Visualization is established as a very useful approach to exploration, analysis, and presentation of large and complex datasets. The human visual system with its powerful parallel information processing abilities is exploited as a broad-band information access channel between the computer and the human mind [19]. However, also this broad channel can get jammed when too much data is to be visualized.

Whereas the treatment of efficiency issues is common in scientific visualization for a long time (in the context of large data visualization), the question of how to cope with millions of data items now also receives increasing attention in information visualization [2]. Many existing techniques, however, are not particularly well-suited for really large datasets. This is especially true for any kind of visualization technique where relatively large amounts of screen space are used to represent individual data items. Examples are star glyphs [16] as well as parallel coordinates [5, 4] where many pixels are needed to represent one single data item – the advantage of providing especially useful insights into multidimensional datasets comes at the cost of a relatively expensive data representation in terms of the necessary screen estate.

Without any special solution applied, the use of parallel coordinates is limited to about a few thousand data items per view. When dealing with hundreds of thousands of data items (or more), standard parallel coordinates become heavily overplotted and the resulting visualization is of little use. To deal with the challenge of accommodating large datasets in information visualization, several approaches have been proposed, ranging from the visualization of selected subsets (*selection*), via the visualization of joint representations of data subsets (*aggregation*), to the visualization of dataset subdivisions (*segmentation*) [7].

Related to the aggregation approach, and also related to similar approaches in scientific visualization [20], image processing [9], and many other fields, the idea of utilizing different levels of details in visualization (in conjunction with a hierarchical data representation) is very interesting [21]. Hierarchical parallel coordinates, for example, are very useful for the visualization of large datasets [3], even though no special treatment of outliers is integrated.

In addition to the new approach of separately handling outliers in the context representation, we propose an *output-oriented visualization design* to enable the scalable visualization of really large datasets. Similar to scientific visualization, where image-order techniques are opposed to object-order techniques [10] to make the complexity of the visualization algorithms predominantly dependent on the size of the visualization output rather than the size of the input data, also in information visualization an output-oriented approach is very promising – when the number of data items to visualize significantly outnumbers the number of pixels to draw, it is useful to design a visualization algorithm according to the question of how a particular pixel is affected by the visualization (instead of considering how a particular data item influences the visualization).

One very useful way to do so (and to effectively exploit the potential of speeding up the visualization) is to convert the data into a frequency-based representation, e.g., in a histogram, or to consider data densities [12]. The technique of parallel sets [8], for example, integrates such a frequency-based data representation with the layout metaphor of parallel coordinates. Also the enormous development of PC graphics hardware and related GPU programming opened the domain for approaches that process the graphical representation of parallel coordinates in a more efficient and more effective way [6]. See Fig. 1 for an example of output-oriented, outlier-preserving focus+context visualization of a large and multivariate dataset as produced by our new approach.

2 Basic Idea

The basic idea of our new approach is the following. We construct an output-oriented data representation of the original data which consists of a finely resolved 2D *bin map* for every neighboring pair of axes in the parallel coordinates view setup. A polyline across all axes, representing one data item, contributes to exactly one bin per bin map, i.e., for every line segment of all the polylines one bin count (in one 2D bin map) is incremented (see Fig. 4 for an illustration). Thereby, a bin map represents the line distribution between two neighboring axes in a frequency-based form.

On the basis of these bin maps, we identify isolated bins which do not well align with the major trends in the visualization. This way, we separate *visualization outliers* (as compared to *data outliers*) for subsequent special treatment. In conjunction with our output-oriented

- *Matej Novotný is with Comenius University, Bratislava, E-mail: mnovotny@fmph.uniba.sk.*
- *Helwig Hauser is with VRVis Research Center, Vienna., E-mail: hauser@vrvis.at.*

Manuscript received 31 March 2006; accepted 1 August 2006; posted online 6 November 2006.
For information on obtaining reprints of this article, please send e-mail to: tvcg@computer.org.

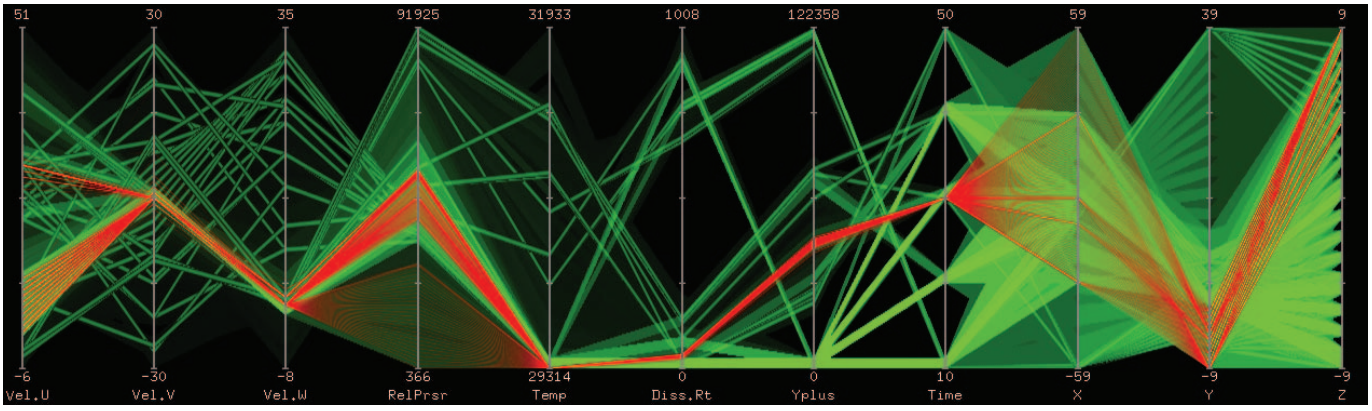

Fig. 1. Outlier-preserving focus+context visualization of a flow simulation dataset (the mixture of two fluids) with the focus on a relatively cold data subset at time step $t = 30s$ (red polylines). The context visualization (in green) reveals, for example, that most flow vectors are aligned with the x-axis (cluster between Vel.V and Vel.W). In addition to the major trends, some interesting outliers are also very well visible.

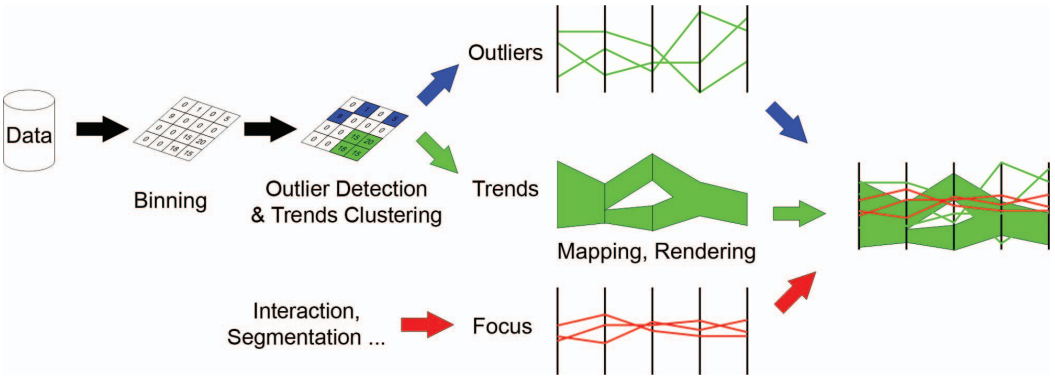

Fig. 2. The workflow for outlier-preserving focus+context visualization in parallel coordinates. After binning, visualization outliers and visualization trends are identified. They are represented separately in the context visualization before they are combined with the detailed representation of the data items in focus.

approach, the consideration of visualization outliers makes special sense – our major goal here is to preserve visualization outliers even though we employ an aggregation scheme to allow for large data visualization (without special treatment, outliers would be smoothed out by the aggregation). Contrarily, in statistical data analysis the data outliers usually are in the focus of research [14, 15] – taking all the data dimensions into account concurrently, isolated data subsets are identified (independent of the visualization mapping).

After isolating visualization outliers, we cluster the remaining (more coherent) contents of the bin maps to extract visualization trends and to enable an aggregated representation of them in the visualization. Due to the preceding isolation of visualization outliers, we obtain sharper clusters with less variance as compared to plain clustering the original data. When composing the eventual visualization, we overlay the aggregated representations of the visualization trends (parallelograms between neighboring pairs of axes) with the non-aggregated representations of the visualization outliers (individual line segments between the axes) and thereby preserve their visual prominence in the visualization.

Since data visualization with parallel coordinates only unfolds its full potential when interaction is supported, i.e., when axes can be reordered/repeated/scaled/flipped/distorted/etc. and when brushing and focus+context visualization is provided, we also incorporate means of interactive visual analysis. Due to the output-oriented character of this visualization approach, we need a sophisticated scheme for deciding when and how to refer back to the original data (in contrast to working with the bin maps only). See also Fig. 2 for an illustration of the workflow of our new approach.

3 Data Binning

Generally, binning is a process during which the original data is converted to a frequency-based representation by dividing the data space into a set of multidimensional intervals – called bins – and assigning to every bin an occupancy value which determines the number of data records that belong to the bin [17].

3.1 Multidimensional Binning

Given a dataset $X = \{ \mathbf{x}_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})^{\mathrm{T}} \mid 1 \leq i \leq m \}$, with m data items of n dimensions, we consider the bounding box $B = [\min_1, \max_1] \times [\min_2, \max_2] \times \ldots \times [\min_n, \max_n]$ so that $X \subseteq B$.

To construct the binned representation of X, i.e., the bin counts $c_{i_1,i_2,\ldots,i_n}$, we consider a decomposition of each of the n sides of B into b_i intervals according to $[\min_i, \max_i] = B_{i,1} \cup B_{i,2} \cup \ldots \cup B_{i,b_i} = [\min_i, \min_i + w_i[\cup [\min_i + w_i, \min_i + 2w_i[\cup \ldots \cup [\max_i - w_i, \max_i]$ with $w_i = (\max_i - \min_i)/b_i$, and define

$$c_{i_1,i_2,\ldots,i_n} = \left| \{ \mathbf{x}_i \mid x_{i,1} \in B_{1,i_1} \wedge x_{i,2} \in B_{2,i_2} \wedge \ldots \wedge x_{i,n} \in B_{n,i_n} \} \right|$$

as the binned representation of X. In total, there are $d = \Pi_{i=1}^{n} b_i$ bin counts $c_{i_1,i_2,\ldots,i_n}$ in this representation. This means that the overall number of bin counts d usually gets very large, even with only moderately large b_i and n – with $b_i = 16$ and $n = 8$, for example, $d = 16^8 = 4.294.967.296$.

The binning transformation preserves the distribution characteristics of the data and replaces it with a frequency-based representation. The exponential growth of the total number of bins, however, causes enormous memory demands when truly multidimensional data is considered, and often even prevents its utilization in such an application

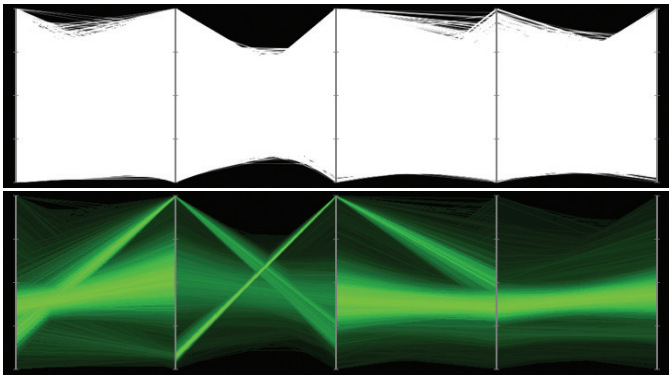

Fig. 3. Remote sense data [18] (interpolated to 100.000 samples) in conventional parallel coordinates (top) and binned to 128·128 bins per bin map (bottom), resulting in a faster and informative display.

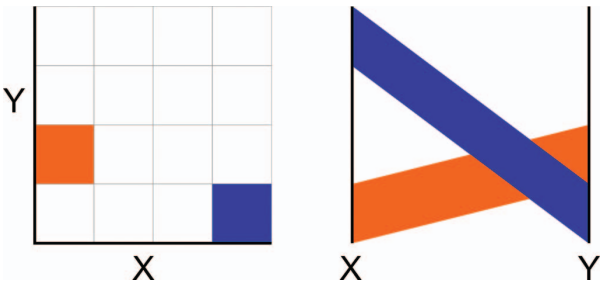

Fig. 4. The bin map (left) containing two non-empty bins (orange and blue). The same bins visualized in parallel coordinates (right).

context. Due to our output-oriented approach to large data visualization, however, we are fortunately not affected by this curse of dimensionality as we will describe in the following.

3.2 2D Binning for Parallel Coordinates

The visualization technique of parallel coordinates is a projection of multidimensional data into a 2D visualization space. According to our output-oriented approach, we apply binning not to the original n-dimensional data but to the 2D visualization space.

For each pair of adjacent axes, representing a pair of dimensions in the data, we bin the particular two-dimensional subspace into $b \cdot b$ bins. The resulting set of bins (for one pair of axes) forms a so called *bin map* and can be thought of as a 2D histogram of the distribution of all line segments between the two axes. For a parallel coordinates setup which is comprised of k axes we need a minimum of $e = (k-1) \cdot b^2$ bin counts to represent the data – even for larger k and b, the total number of bin counts not really gets too large (for $b = 128$ and $k = 16$, for example, half a Megabyte is sufficient to hold all bin maps when using two bytes per bin count). See Fig. 3 for a parallel coordinates visualization based on a binned data representation (lower image) as compared to the traditional approach (upper image).

Accordingly, we can exploit the useful properties of the frequency-based representation to achieve a scalable visualization of large data, but without the disadvantage of the enormous memory demands. The total number of bins is kept sufficiently low so that current computer systems are capable of holding even the complete binned information of all possible two-dimensional subspaces for datasets as wide as 50 dimensions.

Utilizing binning this way results in several major improvements. First, the bin maps and through them the whole approach are well scalable. The size and the number of the bins determine the precision of the aggregation result. The bins are in many cases reusable without the need to re-bin the original data, e.g., to generate a coarser representation from the original fine one. Accordingly, hierarchical or adaptive binning is possible to improve the situation in a data-dependent way.

As a first solution, binning has been implemented to work on the CPU in order to achieve correct and precise results. A GPU-based algorithm would have to face the problem of low color depth of the rendering buffer which would clamp the bin population in large data cases. Modern graphics hardware with high precision floating point textures eliminates this handicap to a certain extent and performing the binning on the GPU is a promising subject to future research.

In previous work [13], we already started to employ binning for the improvement of visualization with parallel coordinates, however without the embedding in an outlier-preserving approach. The parallel coordinates plot consists of k axes placed in the 2D screen space. Each axis A_i represents a certain data dimension j and utilizes a certain mapping transformation that maps the data values $x_{i,j}$ to its screen value $y_{i,j}$. In the sense of the rendering, $y_{i,j}$ is basically the position of value $x_{i,j}$ on the axis A_i. Usually, the default mapping function linearly scales the values to fit onto the axis.

Some of the basic interaction opportunities in parallel coordinates include inverting the axis orientation or adjusting the mapping function so that it zooms to a certain sub-interval within the data dimension. Therefore it is vital to bin the data after the mapping transformation. Otherwise the resulting bin map and the visualization derived from it would not correspond to the standard visual output. By dividing the parallel coordinates plot (k axes) into a set of $(k-1)$ two-dimensional subspaces – each belonging to one pair of adjacent axes – the whole screen can be rendered using only a binned representation.

In addition to 2D binning, bins of different dimensionality also seem to be an interesting option. The increased memory complexity, however, discourages the use of finely resolved multidimensional bin maps. And even for a low number of dimensions (e.g., one or three) the principles of parallel coordinates, based on a 2D decomposition of the original data space, imply that the visual representation of such bins would not be a trivial task.

4 Outlier Separation and Clustering

There are two main reasons to separate outliers from the rest of the data. The first reason is that even though in many application scenarios the outliers are considered as a flaw in the data or as an error in the measurement, there are many other applications where the outliers actually attract a lot of attention and influence the decision process significantly. Network security administrators, for example, inspect an outlier in their data as a potential intrusion. Business experts might interpret outliers in business statistics as an intriguing investment option [11]. Losing an outlier by merging it with the context means obstructing the visual exploration on its way to find interesting areas in the observed data.

Another motivation for handling outliers separately is the quality of data abstraction in focus+context visualization. Every data abstraction aims at effectively balancing simplicity and truthfulness. By introducing outliers to the abstraction, the truthfulness of the abstraction is often significantly decreased and a more complex representation has to be chosen to compensate for that. Fig. 5 shows an example of how outliers might mislead the generation of context for an focus+context visualization and how treating them separately makes the context more coherent with the original data.

4.1 Outlier Detection in Binned Data

Once the data is binned into a two-dimensional density-based representation it can be manipulated in a way that is analogous to two-dimensional signal processing. Operating on such a basis, an outlier can be detected using the following scheme: a low-pass filter is used first, then the result is compared to the original bin and checked for differences. Bins that are emptied after smoothing and quantization to the digital resolution of the bin representation are considered as outlier bins.

A comparison of two outlier detectors – the isolation filter and the median filter – is depicted in Fig. 6. In both cases those bins that have a population below a certain threshold and which also are detected by the filter are marked as outliers (depicted in yellow in the figure). Only

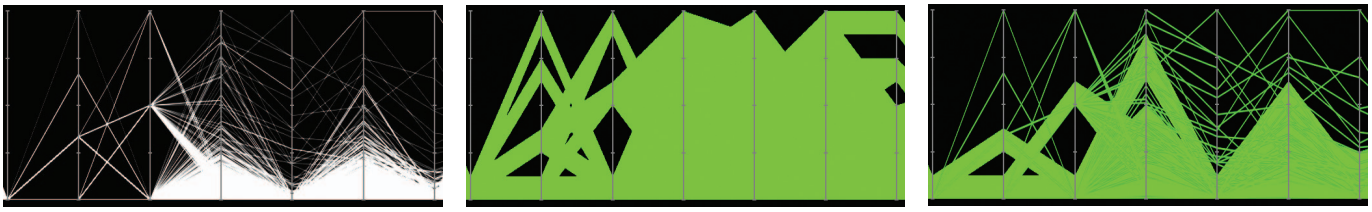

Fig. 5. Without the special treatment of outliers (center), the trends clustering results in a wrongly stretched context representation. Outlier-preserving context generation (right) results in an aggregated visualization which is more truthful to the original data (left).

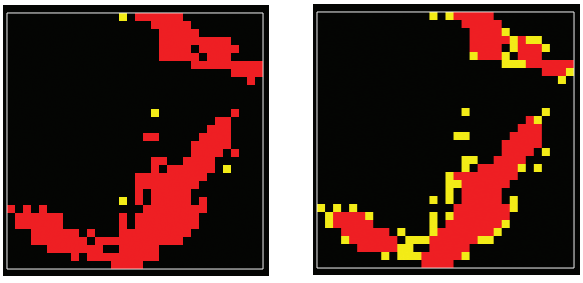

Fig. 6. Comparison of two outlier detection algorithms. Identifying isolated bins (left) is compared to the use of a median filter (right). The outliers are depicted in yellow.

low-population bins are considered to prevent low-variance but highly populated clusters from being interpreted as outliers by mistake. Features of this kind are well visible in the binned visualization and do not need to be treated specially. On the contrary, treating them as outliers would decrease the quality of the visualization since it would remove a significant portion of the whole data from the processing that leads to context generation and/or clustering. During our experiments and in the illustrations here, the population threshold was set between 1 and 10 percent of the maximum population.

The isolation filter has a $3 \cdot 3$ support in the bin map and checks the occupancy values of the 8 bins that are adjacent to the central bin. If the number of empty neighboring bins is above a certain threshold (say 6 or 7) the central bin is declared an outlier. The threshold has to be adequately decreased for the bins on the borders of the bin map (e.g., to 2 for the corners and to 4 for the borders).

Similarly, the median filter computes the median of occupancies of the neighbor bins and if it falls below the population threshold, the central bin is marked as an outlier.

Once the bins are separated into outlier bins and the bins containing the main data, the actual outlier extraction is performed. This is handled in the same way as brushing is handled in parallel coordinates. Those data items that are projected into the outlier bins are extracted from the original data and are represented as polylines in the parallel coordinates.

4.2 Clustering

The clustering approach we introduce in this paper is an interesting extension to the data binning and to the output-oriented processing. Another form of clustering in the screen space of parallel coordinates (utilizing interaction and the grand tour) was introduced by Wegman and Luo [12]. Our approach, however, does not require user interaction and it exploits the theoretical background of statistical data processing as well as data mining.

Clustering as utilized in this paper operates in several steps. First, the binned data is smoothed using a Gaussian filter. As also known from other visualization work [1], smoothing small-scale features (noise, outliers) and compacting the large features (clusters) has a positive impact on the results of the following feature extraction and also of clustering process.

The smoothed approximation of the original data is inspected, starting with the bins with the highest population. The population threshold is iteratively decreased, revealing either new cluster centers or expanding the already existing clusters. The process might run automatically to either search for a desired number of clusters or to finish at a given threshold limit. A reasonable choice is to set the limit to 5–10 percent of the highest occupancy in the bin map. An example of this process is illustrated in Fig.7 and Fig. 8. This process is not exclusively bound to two-dimensional clusters. Clusters of different dimensionality can be created by combining cluster results obtained from different two-dimensional subspaces. Another interesting opportunity is to extract only few clusters from the bin maps but then use the Cartesian product of these clusters to generate a clustering of the entire parallel coordinates plot. Another intuitive way to change the dimensionality of clusters is to merge multiple two-dimensional clusters to a set of one-dimensional ones.

The here described approach allows to use parallel coordinates even for large amounts of data and to utilize the pattern recognition capabilities of domain experts to observe interesting structures in the data and, e.g., to visually detect axes with similar data distribution or to separate the data into structures using only several two-dimensional subspaces in the screen space instead of running a data-oriented clustering on the original large and multidimensional dataset.

5 Context Setup

During the final stage of the proposed approach, the context is composed in an output-oriented way. As the context usually describes the largest portion of the data, it is naturally desired to map the data context to a simplified, aggregated graphical representation. This is achieved using the binned data representation as described above. In our approach, it is possible to reuse the binning which precedingly was used for outlier separation and trend clustering. Alternatively, also a binning with another resolution can be used.

Eventually, every (non-outlier and non-brushed) bin is rendered as a parallelogram connecting a pair of intervals at adjacent axes with its vertexes placed at the respective positions of the minimum and maximum bin borders. To keep the context part visually simple and semantically coherent, only one color is used for all context bins (trends as well as outliers). To discriminate the dense areas from the sparse ones (according to the bin respective counts), the brightness of the bin color is varied. For the actual rendering, all bins of each segment are ordered according to their ascending occupancy and rendered one over another so that the most populated bins are rendered on top of the others using the brightest color.

This approach compensates for disregarding semi-transparency when rendering the parallelograms. It is popular and often also useful to utilize semi-transparency when visualizing a lot of data items. However, for the purposes of generating a visually unobtrusive context in focus+context visualization, the comparably large number of visual attractors as caused by overlapping translucent segments is not a desired effect.

After the context is generated from the core part of the data, the outliers are rendered using the standard parallel coordinates technique, i.e., by drawing polylines for all the outliers. Their color is chosen to be the same as the one of the context trends, so that the outliers are not mistaken for focus items (also rendered in full detail using the usual polylines representation, but in a different color).

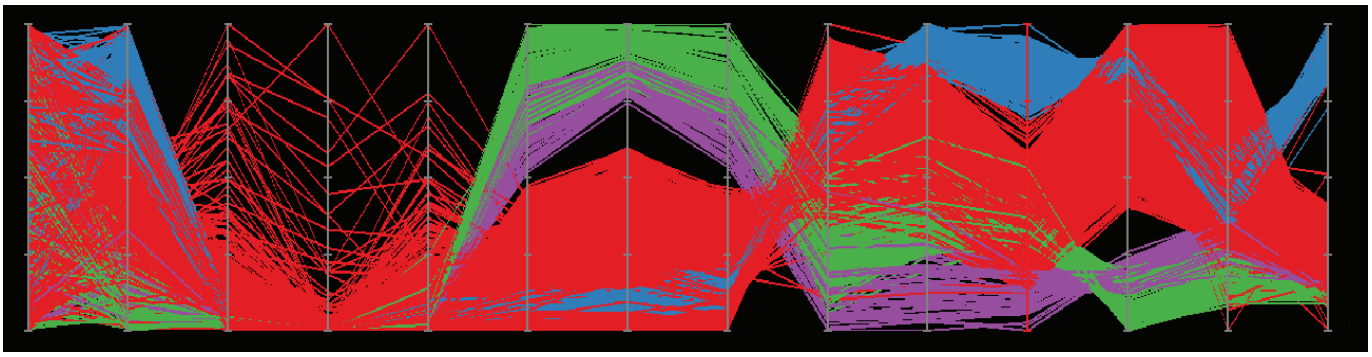

Fig. 7. Polyline coloring in parallel coordinates which represents the result of clustering of the 64·64 bin map between the 11th and 12th axis (in red) as illustrated in Fig. 8.

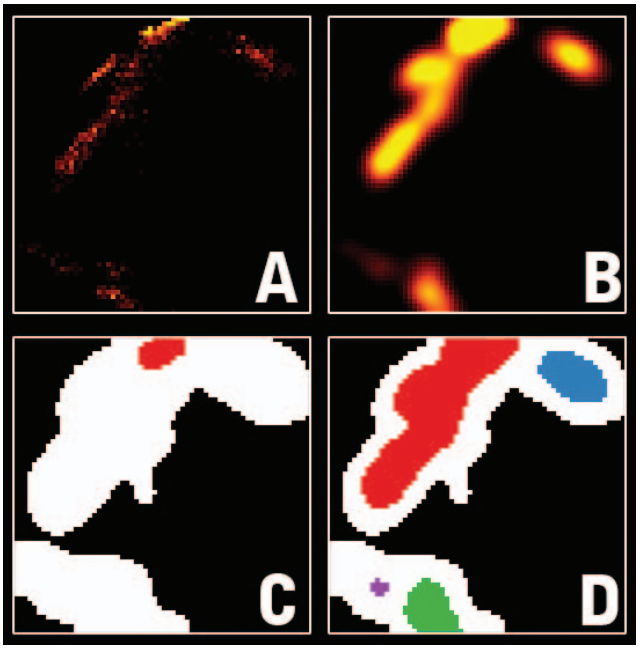

Fig. 8. An illustration of the clustering process which leads to the visualization in Fig. 7. Starting out from a $64 \cdot 64$ bin map (A), first Gaussian smoothing is applied (B) to focus on the major trends in the visualization. Starting from the bins with the highest occupancy, the cluster regions are grown (C,D) by successively including neighboring bins with lower and lower occupancy values until a certain threshold is reached (50% in C and 10% in D).

By design, the context composition (as described above) results in a data representation which only is moderate with respect to its visual conspicuity – by definition, the context visualization should not attract too much attention on the user side (which, of course, should be aligned with the data items in focus, instead). Still, it need not be required to render the context at its lowest possible resolution. Instead, by adjusting the level of details in the context, the user can optimize the precision of the context with respect to his/hers actual needs. Possibly it is also possible, to make this decision (of how finely to resolve the context composition) dependent on overall data properties and/or visualization properties, i.e., measurable characteristics of the resulting visualization, but up to this point we did not investigate this question in more detail.

6 Demonstration and Results

The implementation of our new approach has been done with large multivariate data in mind. Output-oriented rendering is embedded into the application in several ways. Moreover, the demonstration also yields ideas for future research such as, e.g., improving the clustering in screen space or extending the binned data representation.

6.1 Output-sensitive Implementation

To employ the concept of output-sensitive visualization in the most efficient way, three modifications have been done to the original parallel coordinates plot (in addition to the binning-related modifications as presented above), which proved to be very useful, not only for the parallelogram rendering, but also for the polyline rendering. The goal was to retain as much of reusable graphical information as possible and to cut down on unnecessary rendering efforts and expensive data-to-screen mappings.

Layers – The parallel coordinates plot is divided into several layers that contain different portions of the visualization. These portions are focus, context (without outliers), outliers, original polyline representation and clustering results. Different combinations of these layers can be generated by the use of alpha blending, of course, without the need to re-render any of them. Additional layers, such as a data segmentation or details on demand, could be added in, as well.

Segments – To improve the interaction capabilities, the screen is divided into rectangular segments, one between each pair of two adjacent axes. This tightly relates to the concept of 2D binning and creates an effective framework in which only those segments have to be re-rendered which actually change due to interaction.

Rendering to texture – The rise of advanced GPU-programming advises the use of textures in the context of GPU-based visualization algorithms. Final as well as intermediate results are stored in textures and used for following computations or rendering. Similar to other recent approaches in information visualization [6], we also utilize textures to improve efficiency. In the presented prototype, each segment is first rendered to a texture to save the rendering results for future use and then the texture is rendered to the screen.

In total, the parallel coordinates plot is divided into an array of $n_{\text{layers}} \cdot n_{\text{segments}}$ textures. Only those textures are re-rendered during interaction which actually change, all others are re-used.

6.2 Fast and Informative Rendering

The purpose of focus+context visualization is to put fine-scale details about a selected portion of the data into the focus of visualization while still preserving an overview of the rest of the data to provide context information. The solution presented here enriches the context with additional information while keeping the nature of the context simple and easily readable. Therefore more relations between the data in focus and the context can be observed.

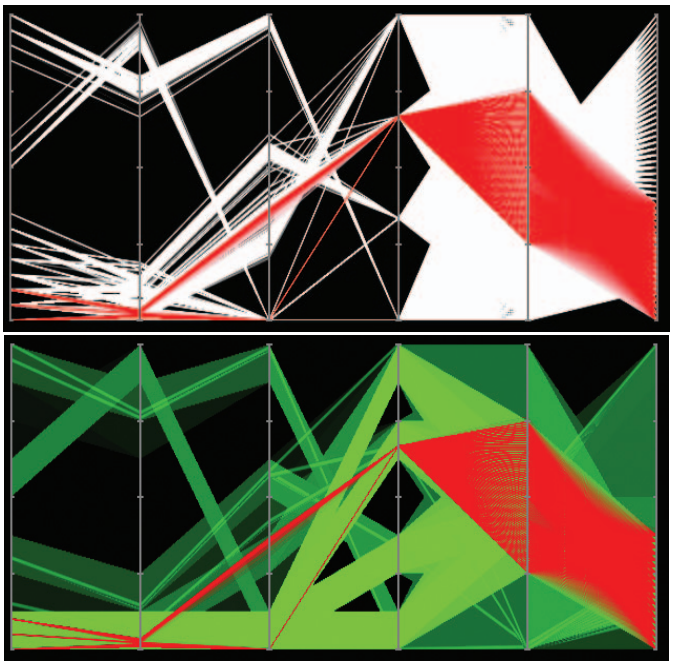

Fig. 9. Standard focus+context visualization in parallel coordinates (top) and outlier-preserving focus+context visualization of the same data (bottom). The lower picture reveals more information about the internal structure of the data context and also about the relation of the focus to the context.

An illustration of the contribution can be seen in Fig. 9. The top picture shows a standard parallel coordinates visualization. The data in focus (depicted in red) is drawn over the rest of the data. Many parts of the plot are heavily overplotted. It is hard to decide on their intrinsic nature. By changing to outlier-preserving focus+context visualization (lower image), a more informative view is produced and specific relations are clarified between focus and context. We can see, for example, that the data items in focus are eccentric with respect to the main trends in the data (see axes 3 and 4). In addition to the advantage of generating more informative views, also there is less stress on both the visualization computation side as well as on the user perception side.

6.3 Information Increase

Due to overplotting in parallel coordinates, when large datasets are visualized, the standard approach quickly reaches its capacity. Also, it easily happens in such situations that the visualization misleadingly draws the attention of the user to the areas with high graphical volume, i.e., the overplotted context areas. Using intelligently aggregated information instead allows to reveal the true nature of the data in a better way.

Although seemingly paradoxical, a lower level of detail can indeed provide a higher level of information in the visualization as also demonstrated in Fig. 10. Even though semi-transparency was used to render the original line-based display (left), the resolution of the alpha channel was soon exhausted. Therefore the area of high values between the first and second axis draws a lot of attention. However, the outlier-preserving context, as generated from the binned data (right), reveals that most of the data records reside in the lowest portions of the first axis. Also, what seems to be an even distribution on the third axis in the line-based display turns out to be an uneven distribution when displayed using the binned data.

Another interesting solution to deal with the low resolution of the alpha channel was presented by Johansson et al [6]. Using a high precision texture as the primary rendering target and then to apply various transfer functions to the texture subsequently offers effective ways to observe different density subranges.

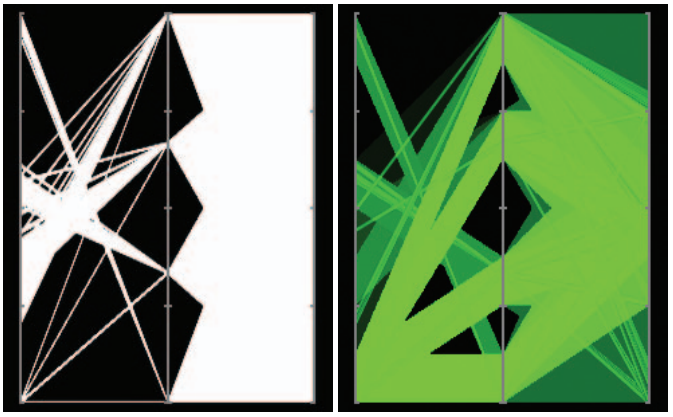

Fig. 10. In the case of large data visualization, outlier-preserving context representation (right) reveals more information about the internal structure of the data than standard, polyline-based rendering, even when semi-transparency is employed (left).

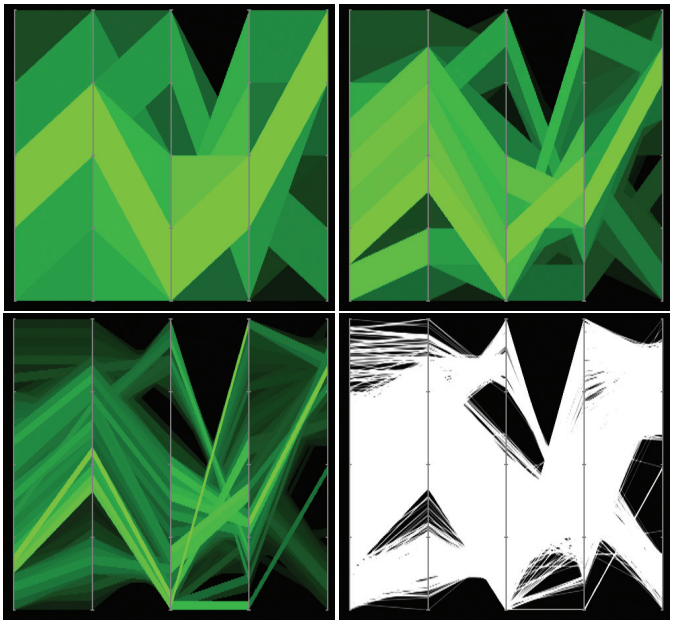

Fig. 11. Different levels of detail, when rendering the context, are compared to standard, polyline-based rendering.

6.4 Variable Context Precision

The context becomes much more flexible once we introduce binning (or data abstraction in general) to focus+context visualization. The context can either be depicted in the usual visually undemanding form, but it can also provide additional information for almost no costs with respect to processing or visual investments. Fig. 11, for example, illustrates how different levels of detail influence the visual appearance of the context visualization. The top left image shows a possible trend. The brightest areas are those with the highest occupancy values. When decreasing the size of the bins, the crude and large structure breaks up into finer ones, revealing new information about the data context.

6.5 Performance Considerations

In interactive exploration of large datasets through visualization, it is crucial to steer the performance of the application in a way so that the display updates in less than 100 or 200 milliseconds. In the case of output-oriented rendering, and in the form as it is introduced in this paper, it is relatively easy to achieve interactive frame rates.

In our prototype implementation, the binning is done once at the

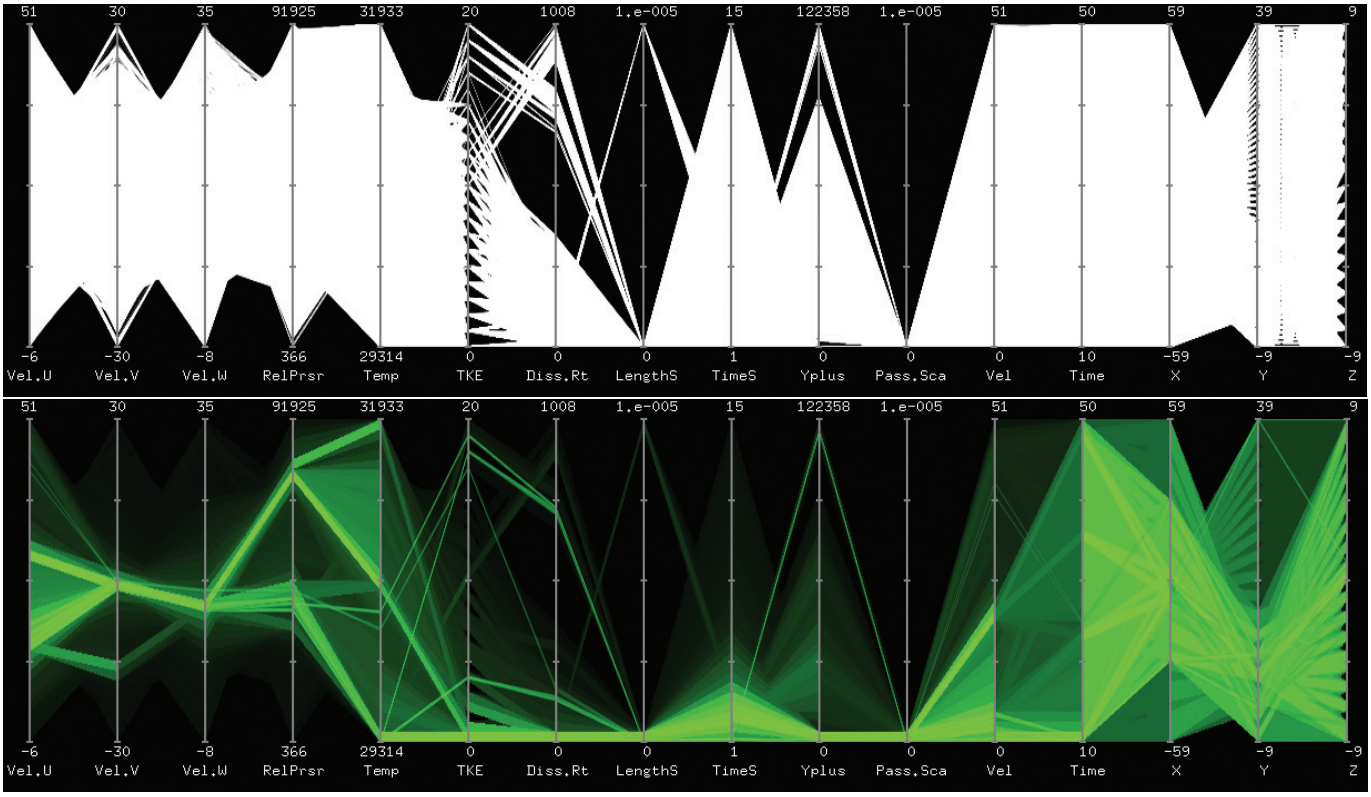

Fig. 12. Output-oriented, outlier-preserving focus+context visualization allows to render more than three million data items into a parallel coordinates plot. Apart from the fact that the binned representation is much more clearer and informative than the standard plot with all the polylines (even when employing semi-transparency), it also renders interactively.

beginning of the process in a fine way (256·256 bins per bin map). This resolution is precise enough to use it as source for successive binning when creating a coarser context representation, or when performing simple axis operations like zooming or panning. The initial binning operation is the most costly operation in the whole framework. The largest dataset, which we tested with our setup (AMD Athlon64 3200+, 2 GB RAM, nVidia GeForce 6600GT), consists of more than 3 million data records in 16 dimensions. Even though the binning of this very large dataset takes a reasonable time (≈70 seconds), the binned data then is rendered instantly. In contrast to that, the standard polyline-based rendering without any pre-processing takes about 3 minutes to accomplish (one rendering). This yields a 2.5 times faster rendering performance, even when also including the preprocessing time.

This implies that even though binning is a very time-consuming process (especially when it comes to large multidimensional data), it is nevertheless a very clever investment as compared to the time taken by rendering the heavily overplotted display over and over. Once the binning is done, the rendering of the bins is accomplished in real-time, also producing a clearer and more informative display. The memory requirements, on the other side, are reasonable. A detailed bin map (256·256 bin counts) takes up 256 KB of memory when four bytes are used per bin count. A dataset with 50 dimensions requires approximately 300 MB to store the bin maps for all possible axis-axis combinations. The binned data can often be reused and the original data has to be accessed only in certain special cases, e.g., after drastic changes to the axis mapping functions (to assure correct binning) or when rendering the data items in focus. The focus contains the full dimensionality of the contained records and therefore the original data have to be used.

Also brushing has to be performed in a standard way. Although there are ways of preserving the inverse mapping between bins and their data members, they are memory-demanding and the actual decision between a fast and a space-efficient solution is up to the particular implementation.

7 Conclusions and Future Work

The visualization approach as presented in this paper shows that output-oriented methods are promising to improve the visualization of large and multivariate data. By combining this approach with data abstraction, namely with binning at different levels of detail, an efficient way of modifying standard parallel coordinates is possible. This advanced focus+context visualization treats outliers separately from the rest of the data for two reasons: (a) to generate a more compact representation of the major trends in the data (not distorted according to the accommodation of outliers) and (b) to prevent outliers from getting lost inside the context (due to smoothing effects in the course of trends clustering).

The resulting visualization is capable of showing the context at different levels of detail while leaving enough visual resources for the outliers and for the data items in focus. Thanks to the density-based, binned data representation the visualization performance does not depend on the size of the input data. Thus large datasets can be explored at interactive frame rates and even new interaction options are enabled, e.g., changing the level of detail of the context. To the best of our knowledge, this is the first time that datasets with several millions of multivariate data items are swiftly visualized in parallel coordinates (after some considerable preprocessing, of course) –see Fig. 12 for a respective sample result.

The density-based information in parallel coordinates allows for interesting screen-oriented data tasks such as clustering or the approximation of the original real world model. Many of them are very demanding and complex in their original data-oriented form. But a scalable and simple representation makes them available even for large datasets and interactive exploration. Some of them are addressed in this paper and, thanks to the promising results, there are many inter-

esting options for challenging future research, including possible improvements of the outlier detection process or hierarchical clustering in the aggregated information. Also the exploitation of the modern graphics hardware might yield significant performance improvement to the binning stage.

Acknowledgments

This work is part of the visualization research at the VRVis Research Center in Vienna, Austria (www.VRVis.at), which partly is funded by an Austrian research program called K*plus*. This research also was supported by the VEGA grant 1/3083/06.

The authors would like to thank to Jürgen Platzer and Prof. Peter Filzmoser for statistical background information on outlier detection and clustering. Thanks also go to Harald Piringer for many good ideas and suggestions. Additionally, we also thank Michael Wohlfart for helping with the final version of this paper. The simulation dataset shown in this paper is courtesy of AVL List GmbH in Graz, Austria.

References

[1] D. Bauer and R. Peikert. Vortex tracking in scale-space. In *Proc. of the Joint Eurographics – IEEE TCVG Symposium on Visualization*, pages 140–147, 2002.

[2] J.-D. Fekete and C. Plaisant. Interactive information visualization of a million items. In *Proc. of the IEEE Symposium on Information Visualization*, pages 117–124, 2002.

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

[4] A. Inselberg. Multidimensional detective. In *Proc. of the IEEE Symposium on Information Visualization*, pages 100–107, 1997.

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

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

[7] D. A. Keim and H.-P. Kriegel. Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge Data Engineering*, 8(6):923–938, 1996.

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

[9] W. G. Kropatsch, H. Bischof, and R. Englert. *Digital image analysis: selected techniques and applications*, chapter Hierarchies, pages 211–230. Springer, 2000.

[10] B. Lichtenbelt, R. Crane, and Sh. Naqvi. *Introduction to volume rendering*. Prentice-Hall, Inc., 1998.

[11] R. D. Martin. Trellis displays for deep understanding of financial data. *Financial Engineering News*, 3, 1998.

[12] J. J. Miller and E. J. Wegman. *Construction of line densities for parallel coordinate plots*, pages 107–123. Springer-Verlag New York, Inc., 1991.

[13] M. Novotný. Visual abstraction for information visualization of large data. In *Proc. of the Central European Seminar on Computer Graphics*, pages 41–48, 2004.

[14] P. J. Rousseeuw and K. Van Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41:212–223, 1999.

[15] R. J. Rousseeuw and A. M. Leroy. *Robust Regression and Outlier Detection*. Wiley and Sons, 1987.

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

[17] B. W. Silverman. *Density estimation for statistics and data analysis*. Chapman and Hall, 1986.

[18] http://www.liacc.up.pt/ml/statlog/datasets.html.

[19] E. R. Tufte. *Envisioning Information*. Graphics Press, 1990.

[20] M. Weiler, R. Westermann, Ch. Hansen, K. Zimmermann, and Th. Ertl. Level-of-detail volume rendering via 3d textures. In *Proc. of the IEEE symposium on Volume visualization*, pages 7–13, 2000.

[21] P. Ch. Wong and R. D. Bergeron. Multiresolution multidimensional wavelet brushing. In *Proc. of IEEE Visualization*, pages 141–149, 1996.