Eurographics/ IEEE-VGTC Symposium on Visualization 2008
A. Vilanova, A. Telea, G. Scheuermann, and T. Möller
(Guest Editors)

Volume 27 (2008), Number 3

Illustrative Parallel Coordinates

K. T. McDonnell[1] and K. Mueller[2]

[1]Department of Mathematics and Computer Science, Dowling College, NY, USA
[2]Center for Visual Computing, Stony Brook University, NY, USA

Abstract

Illustrative parallel coordinates (IPC) is a suite of artistic rendering techniques for augmenting and improving parallel coordinate (PC) visualizations. IPC techniques can be used to convey a large amount of information about a multidimensional dataset in a small area of the screen through the following approaches: (a) edge-bundling through splines; (b) visualization of "branched " clusters to reveal the distribution of the data; (c) opacity-based hints to show cluster density; (d) opacity and shading effects to illustrate local line density on the parallel axes; and (e) silhouettes, shadows and halos to help the eye distinguish between overlapping clusters. Thus, the primary goal of this work is to convey as much information as possible in a manner that is aesthetically pleasing and easy to understand for non-experts.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Display algorithms I.3.3 [Computer Graphics]: Interaction techniques

1. Introduction and Related Work

During the mid-1980s and early 1990s Dimsdale and Inselberg [ID90] introduced a technique for visualization of multidimensional data they called *parallel coordinates* (PC). In this approach, each dimension is drawn as a vertical (or horizontal) line, and each multidimensional point is visualized as a polyline that crosses each axis at the appropriate position to reflect the N-D position. As can be seen in Figure 1, the parallel coordinates methodology facilitates the 2D rendering of very complex datasets in a single image.

The technique suffers from a few shortcomings, which have been addressed by numerous researchers over the years. For example, PC plots tend to be very cluttered, with polylines crossing and overlapping each other. One popular way to address this problem is to use clustering of the dataset [WL97], such as k-means clustering [Mac67], to group nearby N-D points into a single representative N-D point. The clusters themselves can be drawn as heavy polylines on top of the PC plot, and the polylines can be assigned colors to shown their memberships in the mutually disjoint clusters. Other approaches employ multiresolution techniques and impose a hierarchical structure on the data [FWR99]. Algorithmic approaches, like dimension reordering [PWR04], restructure the datasets in an automatic or semiautomatic manner so as to minimize clutter. Brushing [FWR00] is a useful technique for minimizing clutter that permits one to manually omit portions of the data during rendering. Several research teams [NH06, ED06] have also successfully used various focus+context approaches to reduce clutter and to enable the analyst to glean insights into extremely dense datasets.

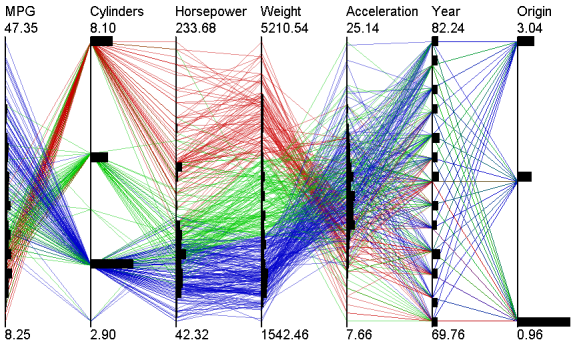


Figure 1: *Traditional parallel coordinates visualization (color-coded by cluster) of the 392-point, seven-dimensional "cars" dataset. Point distributions along axes are given by histogram bars. All datasets visualized in this paper are courtesy of the XmdvTool home page (davis.wpi.edu/∼xmdv).*

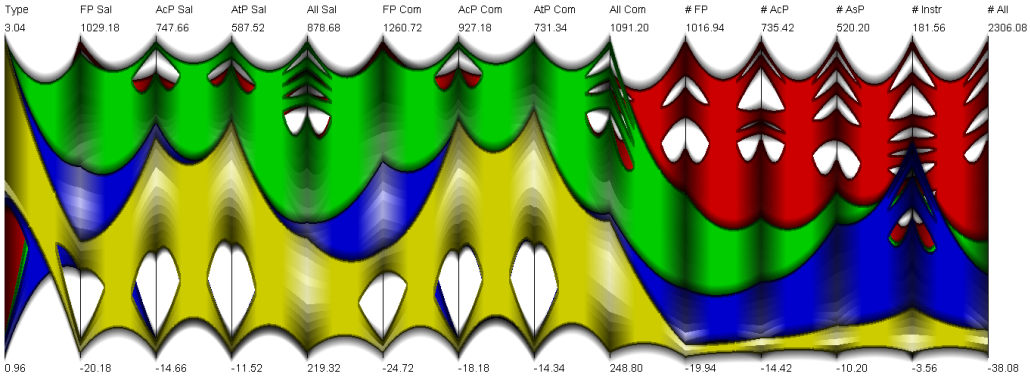


Figure 2: *An example of illustrative parallel coordinates (IPC) showing some of the major features of our new visualization approach. Shading and opacity are used extensively in IPC to convey information.*

Density-based approaches [WL97, JLC07, MW91], including those that employ transfer functions [JLJC05], can also be very helpful in achieving a high-level view of the distribution of the data and can reveal features that may be obscured by numerous overlapping line segments.

Unfortunately, PC plots often do not clearly convey the distribution of the data on each axis. That is, with so many overlapping lines it can be difficult to discern how dense or sparse are the data points on each dimension. A simple solution to this problem is to overlay a bar chart-style rendering of a histogram on each axis [HLD02]. The heights of the bars indicate how many lines intersect the small region covered by each bucket of the histogram. This approach typically works well in practice, but gives the distribution along the entire axis and not on a per-cluster basis.

1.1. Contributions

In this work, we seek to build on the many successful efforts to enhance parallel coordinates plots and offer a new suite of techniques which we called *illustrative parallel coordinates* (IPC). The overarching philosophy behind IPC is to generate parallel coordinate-style visualizations that convey a large amount of information about the dataset in a relatively small area of the screen. Illustrative rendering techniques comprise a proven strategy for revealing structure in data and have been employed with great success in other sub-domains of visualization, such as volume visualization [RE01, LFP*90]. Hence, our major goal is to produce abstract renderings of complex, multi-dimensional datasets that reveal large-scale structures. It is also desirable to create aesthetically pleasing visualizations that draw the eye to important features of the data, such as the distribution of values and the density of the points. We achieve these goals by devising new illustrative rendering techniques and by improving upon existing parallel coordinate visualization methodologies. The specific benefits of IPC and the contributions of this research are as follows:

- Each data-point can be rendered as a polycurve, i.e., a collection of end-point interpolating B-spline curves. Doing so facilitates the creation of edge bundles [Hol06] and serves to de-clutter the visualization.
- Clusters are visualized as a collection of semi-transparent polygons bounded by spline curves, which show the extents of the clusters and which can be scaled to control the screen area they consume. Higher cluster opacities correspond with clusters containing more points.
- The distribution of the data can be viewed at different levels of detail by displaying the clusters in a branched, tree-like manner.
- A density plot that conveys the distribution of the lines or curves between axes can be used to show correlations between axes.
- The distributions along individual axes are shown as faded quadrilateral strips and provide per-cluster histograms of the dataset for each dimension.
- Silhouettes, shadows and halos not only assist the eye in distinguishing between overlapping clusters, but also provide an interesting artistic effect.

2. Illustrative Parallel Coordinates

Throughout the discussion that follows, we assume that a clustering of the data has been provided. Thus, the focus of this work is strictly on the development of new rendering techniques for parallel coordinate plots. Although we chose to use k-means clustering in our implementation for the sake of speed and ease of coding, other clustering techniques – including hierarchical approaches – could be used instead.

2.1. Notation

In this paper we will use the notation $d_{i,j}$ to refer to the value in dimension i of the j-th data-point, where $i, j \geq 1$. We will assume that the dimension, N, of the dataset is at least 4. The notation $v_{i,j}$ will signify the screen space coordinate of point

$d_{i,j}$ and is computed through a linear mapping (a translation and scaling) from dimension i to a vertical line segment that represents the i-th axis. The notation $v_{i,min}$ and $v_{i,max}$ will denote the respective minimum and maximum values of $v_{i,j}$ over all j. We assume that the region between two adjacent dimensions i and $i+1$ is rendered in its own viewport [NH06]. Each viewport's horizontal axis has the range $[-1,1]$, and the vertical axis has the range $[-1/AR, 1/AR]$, where AR is the viewport's aspect ratio. Thus, -1 is mapped to dimension i and $+1$ is mapped to dimension $i+1$. The notation c_i will refer to cluster i and $c_{i,\mu}$ to the center of cluster i. Note that we are assuming that dimension i is mapped to the i-th axis, although this could be generalized easily to accommodate re-ordered axes.

2.2. Edge Bundling

Building on the hierarchical edge bundling concept [Hol06], we can employ B-spline curves [PT97] in IPC to replace polylines with polycurves and thereby decrease the amount of screen real estate required by the PC plot. Graham and Kennedy [GK03] also proposed using curves instead of polylines in PC plots, but their goal was to aid the eye in tracing data points across the axes. Among Moustafa and Wegman's generalizations of parallel coordinate plots [MW02] is also a set of techniques for curve-based PC plots designed to quantize the data and better facilitate analysis. Our implementation employs end-point interpolating B-splines and features a global tension parameter, β, that ranges continuously between 0.0 and 1.0. Higher values of β result in less curved splines. The control points are computed as a function of the original polyline and the cluster to which the point belongs.

To take an example, when using quadratic curves, the positions of the three control points are given in viewport coordinates as $(-1, v_{i,j})$, $(0, \beta m + (1-\beta)p)$ and $(1, v_{i+1,j})$, where m is the mid-point of $v_{i,j}$ and $v_{i+1,j}$, $p = c_{k,\mu} + \lambda(c_{k,\mu} - m)$, and c_k is the cluster to which p belongs (see Figure 3). The λ factor helps to increase the curvature of the splines, and we found experimentally that values in the range $0.5 \leq \lambda \leq 1.0$ generated the most eye-pleasing results. An example of edge bundles using quartic B-spline curves is shown in Figure 4. We observe that the curve bundling very effectively de-clutters the PC plot.

2.3. Spline-based Cluster Rendering

After finding the clusters, the number of which can be set by the user in our implementation of IPC, we can visualize each cluster as a collection of semi-transparent colored polygons that show the extent of each cluster on each axis. The center of the cluster in N-D space is mapped to the axis through a simple projection. The vertical thickness (height) of the cluster on each axis is controlled through a combination of the standard deviation of each dimension i for a particular cluster ($c_{i,\sigma}$) and a cluster scaling parameter (h) set by the user. The parameter h is useful for shrinking the screen area

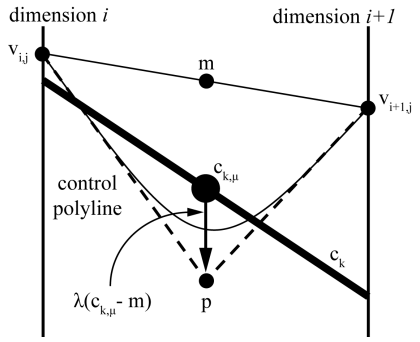


Figure 3: *Each polyline of a traditional PC plot can be transformed into a polycurve to create edge bundles.*

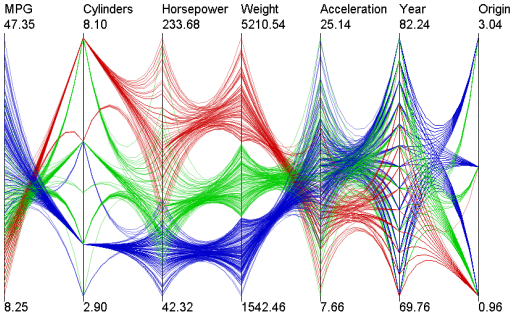


Figure 4: *Edge bundles group polylines into tight groups of polycurves in the "cars" dataset. Curves of the same color are in the same cluster.*

taken up by each cluster, which helps to reveal the structure of clusters obscured by other clusters. Examples can be seen in Figure 5.

Now, given the mean ($c_{i,\mu}$) and standard deviation of the cluster in screen coordinates, the on-screen top extent for dimension i is given by the minimum of $v_{i,max}$ and the screen-space projection of $c_{i,\mu} + hc_{i,\sigma}$. Likewise, the on-screen bottom extent of the cluster is given by the maximum of $v_{i,min}$ and the screen-space projection of $c_{i,\mu} - hc_{i,\sigma}$. Thus, even if a very large scale factor is employed, the top and bottom boundaries of the on-screen representation of the cluster will never extend beyond the actual extents of the cluster. This can be seen clearly in the images of Figure 5 on the axis marked "22s" (third from left). In this dimension, the cluster colored green includes points in a very narrow range.

The curvature of all cluster boundaries can be controlled through a global tension parameter, such as the β mentioned earlier, or an independent one. By modifying the tension of the cluster boundaries, we can reveal clusters occluded by other clusters. The upper and lower boundaries of a cluster are given by end-point interpolating B-spline curves. For each cluster we know the extents of the cluster on each axis in screen space. Taking two adjacent axes at a time we have

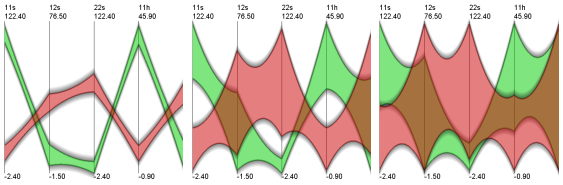


Figure 5: *Scaling the clusters permits one to generate a wide range of visualizations. Shown is part of the "scanbio" dataset for cluster scaling factors* (h) *of 0.5, 3.0 and 12.0, respectively, and with an opacity scaling factor* (α_s) *of 0.5.*

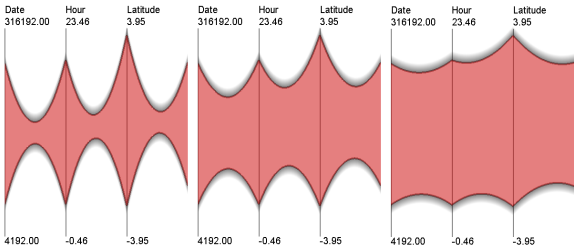


Figure 6: *A portion of the "venus" dataset with spline tension* (β) *values of 0.15, 0.50 and 0.85, respectively.*

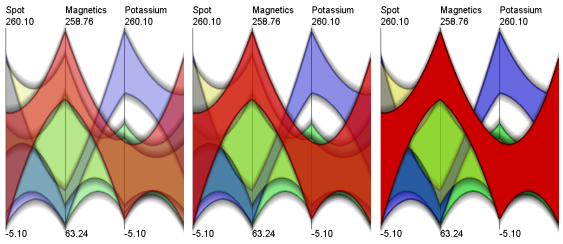


Figure 7: *Visualizations of a portion of the "out5d" dataset with opacity scaling* (α_s) *values of 0.5, 0.75 and 1.0, respectively. Of the four clusters shown, the red one has the greatest size; thus, it has full opacity when* $\alpha_s = 1.0$.

four positions that form a quadrilateral. Call these positions t_l, t_r, b_l and b_r for the top-left, top-right, bottom-left and bottom-right corners, respectively. Also define t_m and b_m as the top mid-point and bottom mid-point as the screen-space mid-points of the line-segments formed by $\overline{t_l t_r}$ and $\overline{b_l b_r}$, respectively. Now, the 2D control points of the upper curve can be given by the triplet $(t_l, \beta t_m + (1-\beta b_m), t_r)$ and those of the bottom curve as $(b_l, \beta b_m + (1-\beta t_m), b_r)$. Examples of cluster boundaries for different values of β are given in Figure 6. Once the control polylines have been determined, the two curves are discretized at the same sampling rate. Quadrilaterals are constructed by connecting pairs of adjacent points in the top spline to pairs of adjacent points in the bottom spline. Specifically, sample points i and $i+1$ in the top spline are combined with sample points i and $i+1$ in the bottom spline to draw the quadrilateral.

We can also employ opacity modulation to encode the size of the clusters, where by "size" we mean the number of points in the cluster. Specifically, given the size, $n(c_i)$ of cluster i, we assign it an opacity of $n(c_i)/\max_j\{n(c_j)\}$. Thus, the cluster with the greatest number of points will have the maximum opacity of 1.0. It is also desirable to give the user the ability to further modulate the opacities of the clusters through a global scaling factor (call it α_s, with $0 \leq \alpha_s \leq 1$) so that the cluster with greatest size need not always have full opacity. Thus, in practice, each cluster is assigned an opacity equal to $\alpha_s n(c_i)/\max_j\{n(c_j)\}$. As can be seen in examples in Figure 7, opacity can play a significant role in both revealing details hidden behind foreground clusters as well as produce very colorful images. One improvement would be to allow the opacity to be modulated on a local level to allow the user to focus in on a particular feature of the dataset. This would be interesting to investigate in a future effort.

2.4. Branched Clusters

A straightforward rendering of the clusters does not provide much insight into the distribution of the dataset along the axes. We have devised a new branching technique that can reveal how densely or sparsely the data are distributed throughout the N-D space. Specifically, we can control the amount of branching by setting a percentage (call it w%) that indicates the minimum width of any branch. For instance, if the percentage is 5% and the height in screen space of the axis is 500 pixels, then the minimum branch height is 25 pixels. As this percentage is decreased, thereby permitting the creation of thinner branches, large branches split into smaller ones. Conversely, as the percentage is increased, small branches merge to form larger ones.

To construct the branches for a given axis with a minimum width of w%, we first sort the values corresponding with the dimension of this axis into an array separate from the original dataset. Then, we iterate through the values starting at the smallest and group those values lying near each other. The boundary between two groups is determined when the distance between two successive values is greater than some threshold. In our implementation we set this minimum distance to be approximately w%, which prevents the branches from touching and helps the eye to separate them. A branch can then be displayed as a pair of spline curves bounding a group of quadrilaterals. Our implementation uses the same discretization sampling rate and rendering approaches described in the previous section.

Examples of branching with different w% values are given in Figure 8. We see how increasing the value of w% decreases the branching factor and permits us to view the dataset at different levels of detail. It is important to note that is possible for a single point (e.g., an outlier) to be mapped to a single branch. One might expect this situation to generate a very undesirable, sliver-like branch with a width of one pixel. However, this possibility is avoided through our use

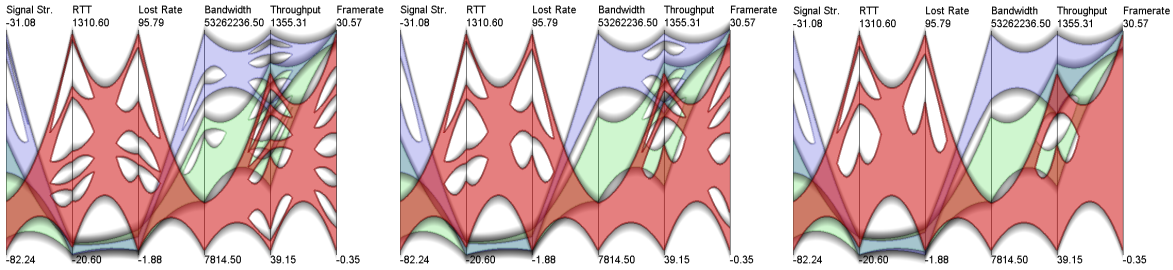


Figure 8: *Branching of the clusters provides insight into the distribution of the data. Shown is the "netperf" dataset with minimum branching widths* $(w\%)$ *of 4.6%, 6.1% and 10.0%, respectively. As w% increases, small branches merge together.*

of the w% parameter, which, by design, forces each branch to have a minimum width significantly larger than a single pixel.

2.5. Silhouettes, Shadows and Halos

All of the visualizations in this paper show IPC plots with silhouettes, which help one to distinguish between overlapping clusters. Silhouettes comprise a well-known technique for clearly marking the boundaries of iso-luminant regions. In IPC, silhouettes consist of B-spline curves drawn on top of the splines that define the boundaries of each cluster. In our implementation they are assigned the same hue as their cluster but a lower brightness so as to contrast them with the cluster's base color. They are very clearly visible in Figure 7.

Shadows can also play a valuable role in distinguishing the boundaries of overlapping clusters. Although all clusters are drawn on the plane, the order in which they are drawn determines how the opacities are blended in the final image. Shadows help to clarify which clusters are on top of which, particularly when the clusters are drawn with high opacity, as shown in Figure 9. They also improve the visual quality of the renderings and make the clusters show up more clearly on the display. We found that they are best used in conjunction with silhouettes.

In our implementation, shadows are drawn by rendering a strip of dark gray quadrilaterals whose opacities fade from 1.0 to 0.0. This provides the desirable soft shadow effect shown in the figures. Near the junction of two branches, it is necessary to taper the shadows. Otherwise, the shadows overlap with the clusters themselves and cause the visually distracting artifacts. In our implementation we taper the shadows down to a thickness of 0.0 near junctions so as the accommodate the rendering of branches very close together. Last, glowing halos [ARS79, IG98] can be created by changing the shadow color to a non-grayscale value; they provide an eye-catching visual effect.

2.6. Faded Histograms within Clusters

Histograms are sometimes overlaid on parallel coordinate plots to give a sense of the data's distribution on each dimension. We have incorporated this idea into IPC and have improved upon it by using fading, shading and warping in drawing the histograms. Even more importantly, our faded histograms provide distributions of the dataset within each cluster on each axis. The count of each entry $b_{i,j}$ ("bucket") in the histogram for axis i is normalized by dividing each bucket's value by the maximum value in any bucket in that axis's histogram. Call this normalized value $b'_{i,j}$. Then, the triple $(b'_{i,j}, b'_{i,j}, b'_{i,j})$ is treated as an RGB grayscale value. Thus, black corresponds with an empty bucket and white with the bucket of greatest magnitude. Each bucket is rendered as a quadrilateral strip with high opacity at the axis itself and tapering off to an opacity of zero away from the axis. We found through experimentation that it is best to visualize the histograms in this approach by displaying the clusters one at a time. Otherwise, the clusters' semi-transparent quadrilaterals overlap, and it can become difficult for one to discern which histogram belongs to which cluster. One way to avoid this problem is to visualize all clusters at full opacity.

To render the histograms, we treat them as bar charts that have been warped according to the spline tension parameter (β) set by the user and by the geometry of the branches. Since the bars are deformed, we will refer to them as "stripes" throughout this discussion. Bucket stripes are drawn on a branch-by-branch basis so as to avoid drawing stripes in empty regions lying near the axes. Within each branch we determine which buckets lie within the branch and draw them sequentially as warped quadrilaterals strips. One end of the stripe is attached to an axis and the other end to a position one-third of the distance to the neighboring branch. Note that the stripes are drawn on both sides of each axis. The x values of the quadrilaterals are spaced equidistantly along the horizontal. The y values are determined through linear interpolation of the y values of the two spline curves that define the upper and lower boundaries of the branches. Use of the splines' y values guarantees that the histogram stripes follow the same contours as the branches, as can be clearly seen in Figure 10. We see that the distributions inside the branches can vary quite dramatically, a fact which is captured quite effectively by the faded histograms. To the untrained eye, these stripes might appear to be a form of diffuse shading of a 3D surface. One possible solution

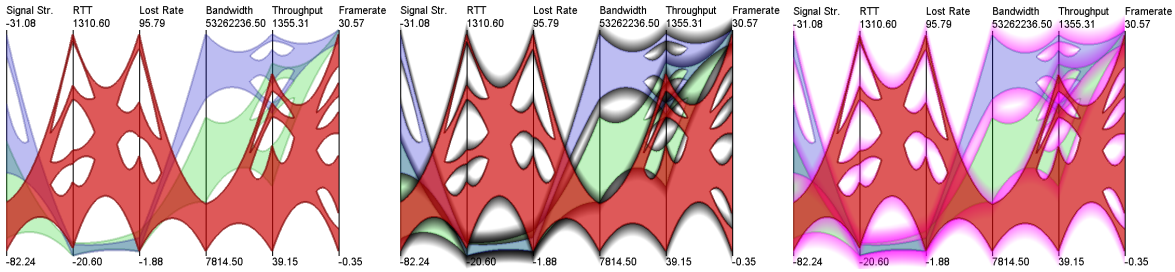


Figure 9: *Shadows help the eye to determine the relative z order of clusters and provide a visually interesting rendering effect. Shown here is the "netperf" dataset without shadows, with shadows, and with magenta halos, respectively. Note that we have drawn the shadow a little darker and wider than usual so that it reproduces well in print.*

would be to put a silhouette around each stripe, but doing so would clutter the image.

2.7. Density Plots

For our illustrative parallel coordinate framework we have devised a technique to generate a density map from the polylines or polycurves so as to give a high-level view of the data distribution and to provide an artistic means of revealing correlations between adjacent axes. For each viewport we render the polycurves of the dataset in an off-screen buffer of the same resolution of the viewport. The curves are rendered in black on a white background. Then, for each pixel at location (i, j) in the buffer we count the number of black pixels in a square-shaped neighborhood of radius r_n of the pixel, including the pixel itself. Given the count $n_{i,j}$ for the pixel at (i, j), we can now define a density value $g_{i,j} = \frac{n_{i,j}}{2r_n+1}$ and generate a density map image. The pixels of this image – except for the white pixels, which represent zero density – must then be inverted so that regions of high density correspond with black and dark gray. That is, each density value $g_{i,j}$ is replaced with $255 - g_{i,j}$, assuming we are using 8-bit textures. To improve the contrast of this inverted image, we then perform a histogram equalization. Next, several applications of a low-pass filter improve the overall density map quality. Optionally, a transfer function can then be applied to map density to color. Finally, the image is texture-mapped onto a quadrilateral, which is then rendered semi-transparently in the viewport. The cluster branches, silhouettes, histograms and other elements are then rendered on top of the density map. An example of a colored density map is shown in Figure 11c. Green indicates low density and red denotes high density. As we can see in the image, there is a strong negative correlation between the rightmost two axes and little or no correlation between the third and fourth axes from the left.

Through experimentation we found that for a viewport of 128×512 pixels, a neighborhood radius (r_n) of at least 10 pixels provides a good starting point for generating the density map. Our implementation then applies a low-pass filter 20 times to the entire image to smooth the map. On the relatively low-end 2 GHz machine on which IPC was implemented, the entire process discussed above usually required approximately one second per viewport.

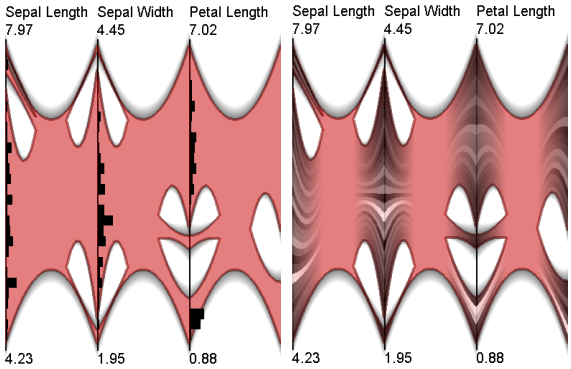


Figure 10: *A portion of the "iris" dataset with histograms rendered in the traditional way and with our new approach. For the sake of this example we grouped the data into a single cluster.*

2.8. Implementation Notes

The images shown throughout this paper were generated by our C++/OpenGL implementation of illustrative parallel coordinates. Each axis is rendered in a separate viewport of 128×512 pixels, which we found provided sufficient width for all the IPC effects to be useful. The development platform was a 2 GHz laptop with 1 GB RAM, and all visual effects, except for cluster identification and generation of the density maps, could be generated at interactive rates for the datasets shown in this paper. We employed very little in the way of GPU acceleration, although it is quite conceivable that certain operations – most notably, the density map generation and spline curve generation – could be accelerated by shifting them to the GPU. The system also contains an FLTK-based GUI for changing the rendering parameters (www.fltk.org). Changes to these parameters are reflected in the display in a fraction of a second, even when all visual

effects are enabled. The drawing of the spline curves can be costly for larger datasets, but even for the "out5d" dataset, which consists of 16,384 points, about one second was required to render all the viewports. Thus, we are very confident that the techniques we have introduced in this paper could be incorporated into existing parallel coordinate visualization systems without imposing a serious performance penalty.

2.9. Caveats

As mentioned in several places, one must be prudent in mixing the different IPC techniques in a single visualization. For instance, the polycurves and density plots work well in conjunction with each other, but do not provide satisfactory results when rendered with the histogram stripes. Second, sometimes it is desirable to use halos instead of black shadows so as to avoid mistaking a dark shadow with a dark histogram stripe. It depends primarily on how many overlapping clusters are visualized at once. Third, rendering the polycurves with the branches occasionally provides unsatisfactory results, particularly when the polycurves are not tightly clustered and/or they are drawn with very low tension.

3. A Dataset in More Detail

In this section we continue to explore the well-known, 7-dimensional, 392-point "cars" dataset. As we saw earlier, Figure 1 gives a traditional parallel coordinates plot. As discussed in Section 2.2, edge-bundling (Figure 4) greatly reduces the clutter and gives one an immediate sense of the distributions of the clusters. The overall structure of the clusters is comparatively difficult to see in the traditional plot. Now, in looking at Figure 1, it is not clear which of the three clusters has the most points. It looks likely that the blue cluster does, but the overlapping lines prevent us from making a definitive call. If we visualize the clusters in branched form (Section 2.4) and enable density-based opacity modulation (Section 2.3), we find that the blue cluster is indeed the largest (Figure 11a).

Although the traditional histograms given in Figure 1 indicate the overall distribution of the dataset along each axis, it would be helpful to understand the distributions within clusters. The traditional plot, even when color-coded by density, cannot effectively provide this information, particularly in the middle portion of the plot. In contrast, the faded histograms (Section 2.6) given in Figure 11b reveal some interesting aspects that are invisible in the traditional plot. For instance, one can notice a bright spot in the red cluster along the "Acceleration" axis (third from right), indicating a locally dense group of points in that cluster. To take another example, the blue cluster along the "Horsepower" axis (third from left) seems to exhibit a fairly uniform distribution of points in the original plot. However, Figure 11b reveals that there are three spots of high density, with the most dense in the center. These two examples alone demonstrate that there are aspects of the data that are virtually impossible to glean from the original PC plot.

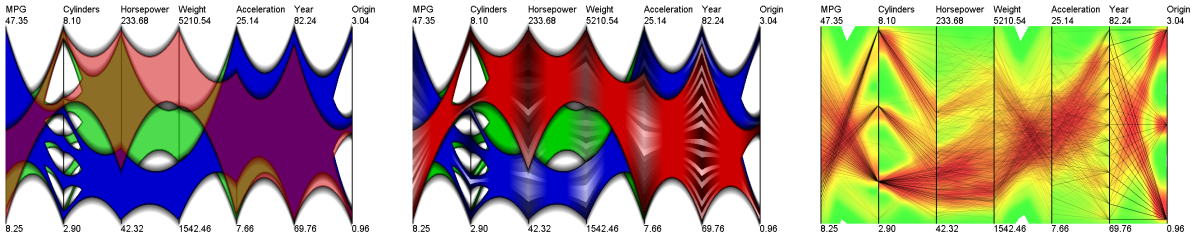
Last, a density plot would be helpful to understand something about the correlations between axes. Figures 1 and 4 provide some preliminary information along these lines. For instance, it would appear that there is a strong negative correlation between "Weight" and "Acceleration," which makes sense in the context of this dataset. Not as immediately obvious is the relationship between "Cylinders" and "Horsepower." The density plot of Figure 11c with overlaid polylines shows a strong negative correlation for high values of "Cylinders," and little or no correlation for lower values. The density plot also provides insight into the last axis, "Origin." The histogram of Figure 1 gives the impression that few points have high values for the "Origin" dimension. The bright red stripe traveling from the lower end of the "Year" axis to the upper end of "Origin" reveals a much higher density of values than the original PC plot and also shows a strong negative correlation between the two axes.

4. Conclusions

We have introduced *illustrative parallel coordinates*, a set of novel rendering techniques for conveying significant and interesting aspects of multi-dimensional data that are also pleasing to the eye and artistic in appearance. Using our branched representation of clusters, an analyst can very easily see the overall distribution of the data across all dimensions. Faded histograms provide additional details by showing the data density within each branch and along each axis. Between axes the density hints give a notion of whether adjacent axes have a positive or negative correlation, or neither. Silhouettes, shadows and halos provide their own artistic touches and assist the eye in distinguishing between overlapping clusters. All these techniques are straightforward to implement and most of the effects generated by our new illustrative methods can be modified at interactive rates on even a low-end PC.

4.1. Future Work

We envision several directions for future research. First, it would be worthwhile to investigate GPU acceleration for the more computationally costly aspects of IPC, such as the spline curve generation and density map creation. Second, it would be interesting to investigate the use of IPC to explore temporal datasets and extend some of the ideas recently presented by Johansson and colleagues [JLC07]. Third, we have not considered the issue of outliers in this work, which should play a role in cluster identification. Fourth, IPC and parallel coordinates approaches, in general, would benefit from increased use of multiresolution techniques to decompose large datasets. Last, a user study would further assist us in gauging the efficacy of IPC and would likely generate additional directions for future work.



(a) (b) (c)

Figure 11: *Several visualizations of the "cars" dataset as part of the case study discussed in Section 3.*

Acknowledgments

This work was supported in part by NIH grant R21 EB004099-01 and NSF grant CCF-0702699. The authors would also like to thank the anonymous reviewers for their helpful comments and suggestions.

References

[ARS79] APPEL A., ROHLF F. J., STEIN A. J.: The haloed line effect for hidden line elimination. In *Proceedings of ACM SIGGRAPH 1979* (1979), pp. 151–157.

[ED06] ELLIS G., DIX A.: Enabling automatic clutter reduction in parallel coordinate plots. *IEEE Trans. on Vis. and Comp. Graph. 12*, 5 (2006), 717–723.

[FWR99] FUA Y., WARD M. O., RUNDENSTEINER E. A.: Hierarchical parallel coordinates for exploration of large datasets. In *Proceedings of IEEE Visualization* (1999), pp. 43–50.

[FWR00] FUA Y., WARD M. O., RUNDENSTEINER E. A.: Structure-based brushes: A mechanism for navigating hierarchically organized data and information spaces. *IEEE Trans. on Vis. and Comp. Graph. 6*, 2 (2000), 150–159.

[GK03] GRAHAM M., KENNEDY J.: Using curves to enhance parallel coordinate visualisations. In *Proceedings of the Seventh International Conference on Information Visualization* (2003), pp. 10–16.

[HLD02] HAUSER H., LEDERMANN F., DOLEISCH H.: Angular brushing of extended parallel coordinates. In *Proceedings of the IEEE Symposium on Information Visualization* (2002), pp. 127–131.

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. on Vis. and Comp. Graph. 12*, 5 (2006), 741–748.

[ID90] INSELBERG A., DIMSDALE B.: Parallel coordinates: a tool for visualizing multi-dimensional geometry. In *Proceedings of IEEE Visualization 1990* (1990), pp. 361–378.

[IG98] INTERRANTE V., GROSCH C.: Visualizing 3D flow. *IEEE Computer Graphics & Applications 18*, 4 (Jul-Aug 1998), 49–53.

[JLC07] JOHANSSON J., LJUNG P., COOPER M.: Depth cues and density in temporal parallel coordinates. In *Proceedings of EuroVis 2007* (2007), pp. 35–42.

[JLJC05] JOHANSSON J., LJUNG P., JERN M., COOPER M.: Revealing structure within clustered parallel coordinates displays. In *Proceedings of the 2005 IEEE Symposium on Information Visualization* (2005), pp. 125–132.

[LFP*90] LEVOY M., FUCHS H., PIZER S. M., ROSENMAN J., CHANEY E. L., SHEROUSE G. W., INTERRANTE V., KIEL J.: Volume rendering in radiation treatment planning. In *Proceedings of the First Conference on Visualization in Biomedical Computing* (1990), pp. 4–10.

[Mac67] MACQUEEN J.: Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability* (1967), pp. 281–297.

[MW91] MILLER J., WEGMAN E.: Construction of line densities for parallel coordinate plots. *Computing and graphics in statistics* (1991), 107–123.

[MW02] MOUSTAFA R., WEGMAN E. J.: On some generalizations of parallel coordinate plots, seeing a million. In *Proceedings of 2002 Data Visualization Workshop (Rain am Lech (nr. Munich), Germany)* (2002).

[NH06] NOVOTNY M., HAUSER H.: Outlier-preserving focus+context visualization in parallel coordinates. *IEEE Trans. on Vis. and Comp. Graph. 12*, 5 (2006), 893–900.

[PT97] PIEGL L., TILLER W.: *The NURBS Book*, second ed. Springer-Verlag, Berlin, 1997.

[PWR04] PENG W., WARD M. O., RUNDENSTEINER E. A.: Clutter reduction in multi-dimensional data visualization using dimension reordering. In *Proceedings of the 2004 IEEE Symposium on Information Visualization* (2004), pp. 89–96.

[RE01] RHEINGANS P., EBERT D.: Volume illustration: nonphotorealistic rendering of volume models. *IEEE Trans. on Vis. and Comp. Graph. 7*, 3 (Jul-Sep 2001), 253–264.

[WL97] WEGMAN E. J., LUO Q.: High dimensional clustering using parallel coordinates and the grand tour. *Computing Science and Statistics 28* (1997), 352–360.