Eurographics / IEEE Symposium on Visualization 2011 (EuroVis 2011)
H. Hauser, H. Pfister, and J. J. van Wijk
(Guest Editors)

Volume 30 (2011), Number 3

Progressive Splatting of Continuous Scatterplots and Parallel Coordinates

J. Heinrich, S. Bachthaler, D. Weiskopf

Visualization Research Center (VISUS), University of Stuttgart, Germany

Abstract

Continuous scatterplots and parallel coordinates are used to visualize multivariate data defined on a continuous domain. With the existing techniques, rendering such plots becomes prohibitively slow, especially for large scientific datasets. This paper presents a scalable and progressive rendering algorithm for continuous data plots that allows exploratory analysis of large datasets at interactive framerates. The algorithm employs splatting to produce a series of plots that are combined using alpha blending to achieve a progressively improving image. For each individual frame, splats are obtained by transforming Gaussian density kernels from the 3-D domain of the input dataset to the respective data domain. A closed-form analytic description of the resulting splat footprints is derived to allow pre-computation of splat textures for efficient GPU rendering. The plotting method is versatile because it supports arbitrary reconstruction or interpolation schemes for the input data and the splatting technique is scalable because it chooses splat samples independently from the size of the input dataset. Finally, the effectiveness of the method is compared to existing techniques regarding rendering performance and quality.

Categories and Subject Descriptors (according to ACM CCS): Probability and Statistics [G.3]: Multivariate Statistics—, Computer Graphics [I.3.3]: Picture/Image Generation—Display algorithms

1. Introduction

Data plots such as histograms, scatterplots, and parallel coordinates are well-known tools in information visualization and descriptive statistics. These techniques are commonly applied in order to analyze trends, correlations, clusters, or distributions. While the data domain may be continuous, the plotting process typically relies on discrete samples that are then rendered using either points or density estimates (such as kernel density estimation). Hence, a discrete model is involved at one stage of the rendering process. In scientific visualization, many datasets are defined on a continuous domain, often within a spatial embedding and one or more data dimensions. Given samples of such continuous data on some type of grid allows continuous reconstruction of in-between data values if an appropriate reconstruction or interpolation scheme is applied. Although the spatial visualization of continuous data with underlying continuous models is a well-known field in the scientific visualization community, continuous density models for histograms [CBB06, SSD*08], scatterplots [BW08], and parallel coordinates [HW09] for scientific data have only recently been published.

While these density models are established now, fast and accurate algorithms for the computation of continuous statistical plots are rare. To fill this gap, this paper presents a new approach to progressive rendering of continuous 2-D scatterplots and multivariate parallel-coordinates plots. To achieve high frame rates for interactive data exploration, a forward-mapping technique inspired by splatting for direct volume rendering is employed to compute the influence of samples to the final image and successively approximate the true representation of the plot. To this end, we present a novel closed-form analytic description of the splatted footprint of Gaussian input kernels for scatterplots and parallel coordinates. We introduce a new progressive refinement algorithm that allows us to obtain initial results extremely fast and hence complements a very useful property of continuous data plots: for many datasets, a small subsample of the full data may provide a good approximation to the final density distribution, making progressive refinement the ideal algorithm for rendering. The paper is completed by performance and image-quality analysis of the GPU implementation of our splatting algorithm.

2. Related Work

Scatterplots and parallel coordinates are common plotting techniques for the visualization, exploration, and analysis of statistical data. A good overview of statistical graphics can be found in [CHU08]. The use of scatterplots in explorative data analysis was promoted by Tukey [Tuk77], whereas parallel coordinates were developed by Inselberg [Ins85]. Recently, a rigorous mathematical description of parallel coordinates was published in a textbook by Inselberg [Ins09]. We refer to this textbook for background information on the mathematics of parallel coordinates. For large data visualization, overdrawing is a severe issue for both types of plots. A common approach to reduce visual clutter uses frequency plots for points in scatterplots—often accompanied by smoothing and kernel estimation techniques [BA97]—and for lines in parallel coordinates [MW91, WL97].

Although statistical data plots were originally introduced using discrete (point-based) rendering, they have also been proven to be useful tools for the analysis of scientific data, which is typically defined on continuous domains within a spatial embedding. Here, the data is normally given on a grid, and the data values at the grid points are interpreted as discrete samples for the construction of the statistical plots. For example, the SimVis system [DGH03] uses histograms and scatterplots in a coordinate-view setup with linking and brushing to explore simulation data. In other examples, 2-D transfer functions are specified with data frequencies visualized in scatterplots [KKH02], and parallel coordinates are used for multidimensional transfer function design [PBM05, TPM05, ZK10]. However, none of these examples considers the continuous domain of the input data field; instead they rely on discrete samples.

To resolve the inconsistency between continuous data model and discrete plotting, Bachthaler and Weiskopf [BW08] introduced a mathematical model that describes the mapping of densities from a continuous input field with known interpolation scheme to the continuous data domain. The same idea can be used to compute continuous parallel coordinates [HW09] by exploiting the point-line duality of Cartesian and parallel coordinates. As a consequence, rendering continuous parallel coordinates always depends on rendering continuous scatterplots, for which an analytic solution is only known for triangulated data with linear interpolation. Bachthaler and Weiskopf further presented an adaptive approach to continuous scatterplots [BW09] that can be used with arbitrary interpolation schemes but assumes constant density distributions within single cells. In contrast, splatting densities allows for flexible density reconstruction kernels, free choice of sample positions, and any type of interpolation scheme. Therefore, we achieve better image quality at comparable rendering speed.

The model of continuous scatterplots is generic and versatile, with various applications and further analysis methods that build upon the construction of those scatterplots. Therefore, any improvement in computational efficiency or quality will benefit those applications. For example, critical line structures can be extracted from continuous scatterplots to obtain further insight in the structure of the data [LT10]. As another example, 2-D joint histograms could, in the form of continuous scatterplots, support information-theoretic flow visualization [XLS10]. Continuous scatterplots may also be interpreted as joint probability density functions that facilitate the identification of mutual information in image registration problems or between multiple 3-D scalar fields in multi-field data analysis [NN11]. Continuous scatterplots also play an important role wherever kernel estimates have to be computed for feature analysis. For example, they could serve as an alternative for the continuous density distribution functions that are used for volumetric transfer function generation [MWCE09]. Finally, the 1-D version of continuous scatterplots—continuous histograms—can be used to examine isosurface statistics for analyzing 3-D scalar fields [CBB06, SSD*08].

Our construction algorithm relies on splatting. We adopt, modify, and extend the splatting method by Westover et al. [Wes89], which was introduced as a forward-mapping algorithm for direct volume rendering. In contrast to raytracing techniques where pixel intensities are computed by mapping the image plane to the data space, volumetric splatting computes the contribution of samples from the data space to the image plane. Since ray integration within a sample's reconstruction kernel is independent of its density, the integral can be pre-computed for a given view direction. The resulting image-plane footprint of the kernel is then used to compute the projected image of all data space samples and only has to be recomputed if the viewing direction changes. For continuous plots, however, the viewing transformation depends on the data, such that footprints have to be computed for every sample individually. Furthermore, the splats for parallel coordinates undergo further transformation according to the point-line duality between Cartesian and parallel-coordinates domains; therefore, those splats differ substantially from those in scatterplots and direct volume rendering.

There is recent work by Zhou et. al. [ZCQ*09] that also applies splatting in the context of parallel coordinates: a Gaussian reconstruction filter is used to adjust the density of discrete samples in a fixed neighborhood of a line segment; by successively accumulating these line splats, a dense field of lines is obtained. It is important to note that the approach of Zhou et. al. [ZCQ*09] reconstructs densities in image space and relies on discrete samples. In contrast, our approach reconstructs continuous densities in the spatial domain and maps them to the parallel-coordinates domain. Therefore, both approaches are independent, both in terms of the type of input data, the shape and behavior of the footprints, the structure of the construction algorithm, and the visualization goal.

Laur [LH91] extended the original volume splatting approach with a progressive refinement algorithm. He uses an octree to subdivide the spatial grid into a hierarchical structure and then draws splats covering the size of a cell in the octree. We employ a similar refinement strategy, although we use arbitrary resampling techniques instead of a subdivision approach.

An approach similar to splatting densities has recently been presented by Feng et. al. [FKLT10] for the visualization of uncertain data samples. In their work, a probability density function is estimated using normally distributed, uncorrelated kernels. Hence, all samples in the scatterplot are represented by scaled Gaussian footprints. In contrast, our model uses gradient information contained in the data to transform the density distribution to both scatterplots and parallel coordinates. Furthermore, densities are derived analytically to allow for the precomputation of splat footprints.

3. Footprint Computation

This section describes the computation of density footprints for reconstruction kernels in the original spatial domain of the data and the respective mapping to scatterplots and parallel coordinates. Note that we use terminology from measure theory and thereby use weighted, non-normalized densities (as opposed to the probability density typically used in the statistics literature). The two main ingredients for the mathematical model of the statistical plots are:

- Density function $s(\mathbf{x}) : \mathbb{R}^3 \to \mathbb{R}$, $\mathbf{x} \mapsto s(\mathbf{x})$, which describes the scalar-valued density over the *spatial domain* of the data set, i.e., the importance of the respective spatial part of the data.
- Map $\tau : \mathbb{R}^3 \longrightarrow \mathbb{R}^n$, $\mathbf{x} \mapsto \tau(\mathbf{x})$, which represents the multivariate data set defined on the same spatial domain as above.

Note that for most information visualization data, there is no spatial domain and thus no concept for s and τ. In typical applications of scientific visualization, τ is given on a grid where each grid point has attached a respective n-tuple of data values. In-between values (away from grid points) are reconstructed by an explicitly or implicitly given reconstruction scheme—typically by trilinear interpolation. For 2-D scatterplots, $n = 2$ data attributes are visualized, e.g., temperature and pressure from a simulation in computational fluid dynamics.

The density s in the spatial domain is typically provided by the user or originates from some kind of external feature definition. If not stated otherwise, a constant density $s(\mathbf{x}) = 1$ can be assumed.

The mathematical problem setting can now be formulated as follows: what are the transformed density functions in the *scatterplot domain* (which is identical to the data domain in [BW08]) and in the *parallel-coordinates domain*? These transformed density functions explicitly depend on the density s in the spatial domain and are implicitly affected by the map τ, which connects from the spatial domain to the other two domains. Figure 1 illustrates the three domains and the example of the mapping of a typical splatting kernel.

We use the following mathematical terminology, similar to previous work [BW08, HW09]. Variables with Latin letters refer to quantities in the spatial domain, e.g., position $\mathbf{x}$ or density s. Variables with Greek letters denote quantities in the scatterplot and parallel-coordinates domains, e.g., ξ for the position in the scatterplot domain and η for the position in the parallel-coordinates domain.

In the following, we first define splats in the spatial domain and discuss the map τ in the context of sampling at the center points of splats (Section 3.1). Then, we restrict the discussion to the transformation of a single splat template, applying this model to continuous 2-D scatterplots (Section 3.2). Finally, we describe the analogous transformation of splats to parallel coordinates (Section 3.3). Here, we restrict ourselves to the bivariate case, $n = 2$. The general case of multivariate parallel-coordinate plots is easily covered by applying the result for $n = 2$ to all pairs of adjacent parallel-coordinate axes.

3.1. Spatial Domain and Data-Set Function

Following the splatting approach for direct volume visualization [Wes89], we represent the density in the spatial domain by a weighted sum of kernels at various spatial locations. Then, the overall density on $\mathbb{R}^3$ is given by

$$s_{\text{overall}}(\mathbf{x}) = \sum_i w_i \, s_i(\mathbf{x})$$

with scalar-valued weights w_i and kernels s_i.

Similar to splatting in volume rendering, the kernels s_i are assumed to be derived from a single template function that is just shifted to different positions. Furthermore, such a template is usually assumed to be spherically symmetric due to reasons of isotropy.

The main idea of splatting is that the template is transformed in a pre-processing step to form a splat. The transformation of the overall density s_{overall} is then computed by overlaying the splats with the same weights w_i. This approach requires that transformation and weighted summation are commutative, which is true for continuous scatterplots and parallel coordinates because both are computed by linear operators.

Now, we consider the concrete example of a 3-D Gaussian kernel in the spatial domain:

$$s_i(\mathbf{x}) = e^{\frac{-\|\mathbf{x}-\mathbf{x}_i\|^2}{k^2}}$$

The kernel s_i is centered at the point $\mathbf{x}_i$ with relative "radius", *bandwidth*, or *smoothing factor* k. The Gaussian ker-

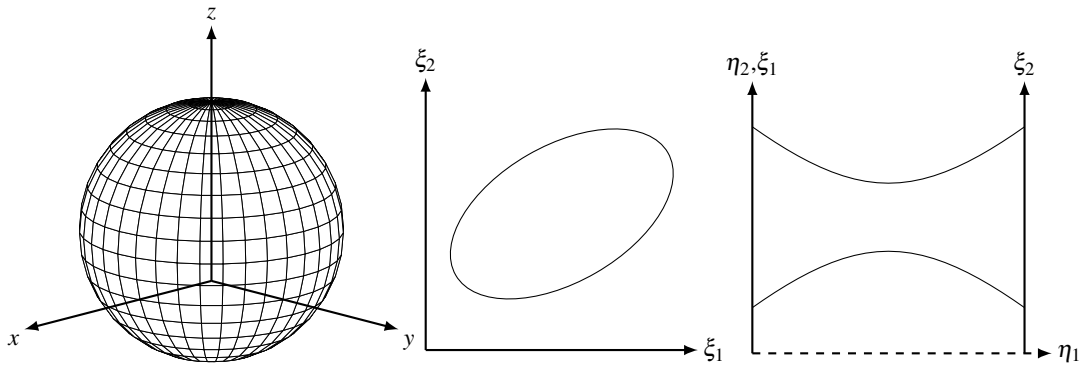

Figure 1: *A spherical reconstruction kernel in the spatial domain (left) maps to an ellipse in the scatterplot domain (middle) and a hyperbola in the parallel-coordinates domain (right).*

nel is popular in volume rendering due to its fast fall-off behavior with increasing distance from the kernel center. We use it for the same reason. Please note that we apply non-normalized Gaussians; normalization is implicitly absorbed by the weights w_i.

The sample positions $\mathbf{x}_i$ may be arbitrarily chosen. In particular, they are independent from positions of grid points of the data set. Typically, the sample positions $\mathbf{x}_i$ are evenly distributed in space, e.g., by putting them on a regular sampling grid or by applying low-discrepancy point sets (see Section 5). The usual case of constant overall density s_{overall} can be implemented by even distribution of sample positions and constant weights w_i.

In the rest of this section, we will only consider a single Gaussian kernel. For simplicity of notation, this kernel is denoted

$$s(\mathbf{x}) = e^{\frac{-\|\mathbf{x}-\mathbf{x}_0\|^2}{k^2}} \tag{1}$$

and centered at $\mathbf{x}_0$.

Furthermore, we assume that τ is a C^1 continuous function. Then, τ can be approximated by Taylor expansion around $\mathbf{x}_0$ up to first order:

$$\tau(\mathbf{x}) = \tau(\mathbf{x}_0) + D_\tau(\mathbf{x}_0)(\mathbf{x}-\mathbf{x}_0) + O(\|\mathbf{x}-\mathbf{x}_0\|^2)$$

where $D_\tau(\mathbf{x}_0)$ is the Jacobian of τ (i.e., the matrix of partial derivatives with respect to spatial locations) as evaluated at $\mathbf{x}_0$. This approximation leads to a linearization of τ around $\mathbf{x}_0$, which is appropriate within a sufficiently small neighborhood around the kernel center, i.e., for sufficiently small kernels. In the limit process of infinitesimally small kernels, the linearized τ converges to the true map. In the remainder of this section, we will work with the linearization of τ. This implies that derivatives of τ are constant. We also assume non-degenerate cases where τ is not constant and its partial derivatives lead to linearly independent gradient vectors.

3.2. Scatterplot Domain

Let us now transform the density function described by the kernel $s(\mathbf{x})$ in the spatial domain to the corresponding density function $\sigma(\xi)$ at position $\xi = (\xi_1, \xi_2)^T$ in the 2-D scatterplot domain. We also call $\sigma(\xi)$ footprint of the splat or, in short, just splat or footprint.

Using Eq. (9) from reference [BW08], the footprint is

$$\sigma(\xi) = \int_{\tau^{-1}(\xi)} \frac{s(\mathbf{x})}{|\text{Vol}(D_\tau(\mathbf{x}))|} \mathrm{d}x$$

where D_τ denotes the 2×3 Jacobi matrix for the bivariate data map τ. The volume measure $|\text{Vol}(D_\tau(\mathbf{x}))|$ is given by the vector cross product of the two gradients of the components of ξ with respect to the spatial domain (see Eq. (10) in reference [BW08]):

$$|\text{Vol}(D_\tau(\mathbf{x}))| = \|\nabla\xi_1 \times \nabla\xi_2\|$$

Since τ is linear, the partial derivatives are constant and the volume measure can be moved outside the integral. In addition, by using the definition of the Gaussian kernel from Eq. (1), we obtain:

$$\sigma(\xi) = \frac{1}{\|\nabla\xi_1 \times \nabla\xi_2\|} \int_{\tau^{-1}(\xi)} e^{\frac{-\|\mathbf{x}-\mathbf{x}_0\|^2}{k^2}} \mathrm{d}x$$

Since τ is linear, the isosurfaces corresponding to isovalues ξ_1 and ξ_2 are planes with normal vectors $\nabla\xi_1$ and $\nabla\xi_2$, respectively. Then, the intersection of the two isosurfaces—identical to $\tau^{-1}(\xi)$—is a straight line. Integration of the Gaussian kernel along this infinitely long line resembles the projection of splats in volume rendering. Therefore, similar to volume rendering, we also retain Gaussian splats in the scatterplot domain. According to the mathematical derivation in the appendix, the density in the scatterplot domain reads

$$\sigma(\xi) = \frac{\sqrt{\pi}}{\|\nabla\xi_1 \times \nabla\xi_2\|} e^{-(\xi-\xi_0)^T E(\xi-\xi_0)} \tag{2}$$

Here, $\xi_0 = \tau(\mathbf{x}_0)$. The matrix $E = \frac{1}{k^2}\left(D_\tau D_\tau^T\right)^{-1}$ is a

2×2 symmetric, positive definite matrix that can be used to parameterize the Gaussian kernel. Following Westover [Wes90], E defines the extents and rotation angle of the screen-space ellipse that is obtained after transformation of a sphere using the Jacobian $D_\tau(\mathbf{x})$ as generalized viewing transformation. The transformation to the standard 2-D Gaussian only depends on the scales S_x and S_y along the main axes of the ellipse as well as the angle θ by which the standard Gaussian is rotated. Due to the symmetry of E, S_x, S_y, and θ are given by the eigenvalues and eigenvectors of the inverse transformation E^{-1}:

$$E^{-1} = k^2 D_\tau D_\tau^T = k^2 \begin{bmatrix} a & b \\ b & c \end{bmatrix}$$

with

$$\begin{aligned} a &= \nabla\xi_1^T \nabla\xi_1 \\ b &= \nabla\xi_1^T \nabla\xi_2 \\ c &= \nabla\xi_2^T \nabla\xi_2 \end{aligned}$$

After calculating the characteristic polynomial of E^{-1} and with $e = \sqrt{(a-c)^2 + 4b^2}$, the scales are computed from the eigenvalues

$$S_x = \sqrt{\lambda_1}, S_y = \sqrt{\lambda_2}$$

where $\lambda_1 = \frac{k^2}{2}(a+c+e)$ and $\lambda_2 = \frac{k^2}{2}(a+c-e)$. If $b = 0$, E^{-1} is a scaling matrix such that $\theta = 0$. With $b \neq 0$, the eigenvectors can be written as

$$\mathbf{v}_1 = (b, \lambda_1 - a)^T, \mathbf{v}_2 = (b, \lambda_2 - a)^T$$

and θ corresponds to the angle of the eigenvectors to the unit vector $(1,0)^T$:

$$\theta = \arccos\left(\frac{b}{\sqrt{b^2 + (\lambda_1 - a)^2}}\right)$$

Now, the footprint of any sample can be transformed to a generic template footprint—usually the standard Gaussian defined by $e^{-\|\mathbf{x}\|^2}$—using a scaling matrix with S_x and S_y multiplied by a matrix for rotation around the z axis with angle θ.

Although the splat computation for scatterplots resembles the one for volume rendering, there is an important difference: the scatterplot-domain splat additionally depends on τ, which is not the case in volume rendering.

3.3. Parallel-Coordinates Domain

Analogous to the previous transformation from spatial domain to scatterplot domain, let us now examine the transformation of density to the parallel-coordinates domain. According to Eq. (7) from reference [HW09], the footprint in parallel coordinates reads

$$\varphi(\eta_1, \eta_2) = \frac{1}{\|\tilde{\mathbf{n}}\|} \int_{\mathbf{g}} \sigma(\mathbf{g}(t))\, \mathrm{d}t \tag{3}$$

where φ is the density of a point $\eta = (\eta_1, \eta_2)^T$ in parallel coordinates with the dual line

$$\mathbf{g}(t) = \frac{\eta_2}{\|\tilde{\mathbf{n}}\|}\mathbf{n} + t\mathbf{n}^\perp \tag{4}$$

in the coordinates of the scatterplot domain. The vector $\mathbf{n}$—the vector that is perpendicular to $\mathbf{g}$—reads

$$\mathbf{n} = \frac{\tilde{\mathbf{n}}}{\|\tilde{\mathbf{n}}\|}, \tilde{\mathbf{n}} = (1 - \eta_1, \eta_1)^T$$

Note that $\mathbf{n}$ and its perpendicular tangent vector, $\mathbf{n}^\perp$, only depend on η_1, whereas the distance of $\mathbf{g}$ to the origin,

$$\frac{\eta_2}{\|\tilde{\mathbf{n}}\|}$$

linearly depends on η_2.

Plugging Eqs. (4) and (2) into Eq. (3) yields the dual footprint of Eq. (2) in parallel coordinates:

$$\begin{aligned} \varphi(\eta) &= \frac{1}{\|\tilde{\mathbf{n}}\|} \int_{\mathbf{g}} \sigma(\mathbf{g}(t))\mathrm{d}t \\ &= \frac{\sqrt{\pi}}{\|\nabla\xi_1 \times \nabla\xi_2\|\, \|\tilde{\mathbf{n}}\|} \int_{-\infty}^{\infty} e^{-(\mathbf{g}(t)-\xi_0)^T E (\mathbf{g}(t)-\xi_0)}\, \mathrm{d}t \\ &= \frac{\pi}{\|\nabla\xi_1 \times \nabla\xi_2\|\, \|\tilde{\mathbf{n}}\|\, \|\mathbf{d}\|} e^{-\frac{\|\mathbf{v}\|^2 + A^2}{k^2}} \end{aligned} \tag{5}$$

with $\mathbf{v} = D_\tau(\mathbf{x})^{-1}(\frac{\eta_2}{\|\tilde{\mathbf{n}}\|}\mathbf{n} - \xi_0), A = \frac{\mathbf{d}\cdot\mathbf{v}}{\|\mathbf{d}\|}$ and $\mathbf{d} = D_\tau(\mathbf{x})^{-1}\mathbf{n}^\perp$. Equation (5) can be derived using a similar approach as for Eq. (2) presented in the appendix.

4. Sampling and Progressive Refinement

The previous section provided the basis for rendering a single splat in the continuous statistical plot. The overall plot is obtained by applying this process to many different splats that densely cover the spatial domain, leading to an appropriate representation of the overall density by accumulating the splats with additive blending. Besides the data values, the partial derivatives of the data are reconstructed at the splat sample so that the position, size, and orientation of the splat can be determined. In this way, the generic footprint template is transformed according to Section 3. Using the inverse transformation, a splat is rendered as a rectangle with the correct texture coordinates and density values.

The generic template footprints for scatterplots are discretized in a 2-D texture during pre-processing. The 2-D standard Gaussian $e^{-\|\mathbf{x}\|^2}$ is sampled on the uniform grid of the texture, which then serves as a lookup table during rendering. The rendering algorithm only needs support for 2-D textures and blending. Therefore, it lends itself immediately to direct and efficient GPU implementation. Due to the rotation $\leftrightarrow$ translation duality between parallel coordinates and cartesian coordinates, the hyperbolic standard footprint from Eq. (5) cannot be sampled to a single 2-D texture, but

can be evaluated efficiently on the GPU using a fragment program.

Conventional discrete scatterplots and parallel-coordinates plots of scientific data are commonly used to visualize the data values attached at the input grid points. For large datasets, however, rendering becomes prohibitively slow hindering interactive exploration of the data. In this case, our splatted continuous plots allow us to trade accuracy for rendering performance by resampling the data at a lower sampling rate. This may be implemented by skipping some of the input data points or by resampling at a few, freely chosen positions on the spatial domain. In contrast, for very small datasets, additional samples may be distributed over the spatial domain to improve image quality. A key observation is that the continuous plots are based on samples that are completely independent from the number and positions of the grid points of the dataset. We recommend making use of this flexibility by employing sampling positions obtained from low-discrepancy sequences [Nie92], such as Halton or Hammersley sequences, because they guarantee even coverage of the spatial domain and, at the same time, avoid aliasing or moiré artifacts from regular sampling.

We further improve the splatting technique by extending it to progressive rendering. Due to the linear superposition of splats, progressively sampled intermediate images can be combined by linear superposition as well. More specifically, we generate a sequence of several, independent continuous plots of low sampling resolution that are then accumulated in a separate image (e.g., an offline rendering target on the GPU). To guarantee mass conservation of the transformed densities (see details on mass conservation in [BW08,BW09, HW09]), single images I_1 and I_2 are composited using the over operator [PD84] and a fixed value for α:

$$I = \alpha I_1 + (1-\alpha) I_2 \quad \text{with } 0 \leq \alpha \leq 1$$

If both intermediate image observe mass conservation (i.e., each has the same overall mass M as accumulated from the densities or all pixels), then the blended image is guaranteed to have identical mass as well because mass is also subject to alpha blending: $\alpha M + (1-\alpha)M = M$. Here, α is a parameter controlling the contribution of densities from a single splatting step to the final image.

The footprints from Section 3 further depend on the choice of the smoothing parameter k, which is a measure for the "radius" or bandwidth of the Gaussian kernel in the spatial domain. This parameter is transported to the scatterplot and parallel-coordinate domains, where it affects the extents of the footprints to be drawn. On the one hand, large kernels provide a better coverage of, and higher overdraw on, the reconstructed spatial area and thus allow us to reduce the sampling resolution. On the other hand, large splats introduce a large error to the overall density, resulting in potentially overblurred images. Furthermore, large splats have a negative impact on rendering performance as the size of the primitives that have to be processed increases. Analogously, small kernels produce a more accurate, but potentially non-smooth sampling of the density distribution.

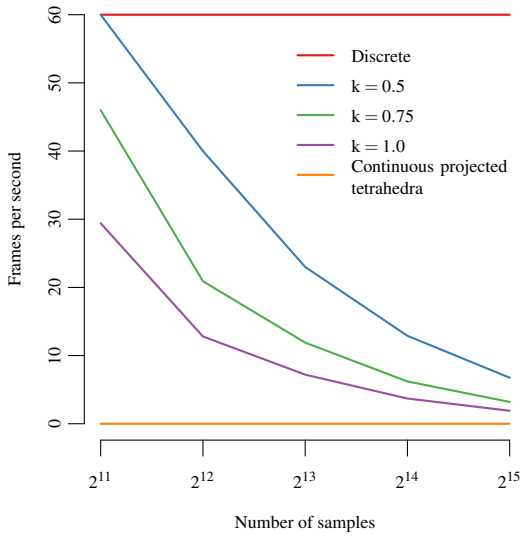

Figure 2: *Rendering performance for splatted continuous scatterplots of the "bucky ball" dataset depending on splat size and sampling frequency. From the plot, a linear dependancy between sampling resolution and frames per second can be concluded. Note that, for comparison, the performance of traditional discrete scatterplots rendered with point-size one and of continuous scatterplots rendered with the projected tetrahedra algorithm are included. The traditional scatterplots are limited to 60 frames per second by the frame refresh rate of the test-hardware. In contrast, the framerate of continuous scatterplots lies well below one frame per second.*

To describe the relative smoothness of the continuous statistical plots, we define the *coarseness*

$$c = \frac{S}{kN} \tag{6}$$

where N is the number of splats, S is the total number of grid points and k is the bandwidth of the Gaussian reconstruction kernel in the spatial domain. As a reference for uniform grids with spacing one, the coarseness equals one if all grid points are sampled using a Gaussian kernel with $k = 1$. According to Eq. (6), the relative coarseness increases with decreasing number of splats and with decreasing splat size. This can be used in addition to alpha blending to obtain a progressively refining image by decreasing both the blending factor and the kernel size in every rendering frame.

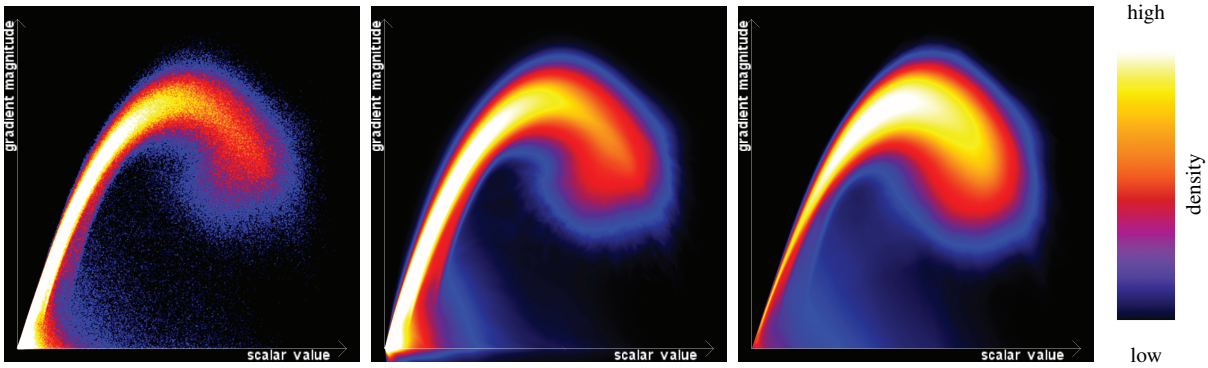

Figure 3: *The "bucky ball" dataset rendered using three different approaches for scatterplots: Discrete scatterplot (left), splatted densities (middle), and the original continuous approach using projected tetrahedra (right). Density is represented by color, where black denotes low density and white denotes high density. Although the splatted and discrete plots use exactly the same samples, the splatted image provides a better approximation to a continuous density distribution. Please note that the plots are not supposed to be identical, as the original approach uses a piecewise linear interpolation model on a tetrahedral grid, whereas the discrete and splatted versions are based on piecewise trilinear interpolation on a uniform grid. Nevertheless, the plots show an almost identical density distribution, where the prominent arc is an indicator for a material boundary.*

5. Results

In this section, the results obtained with splatting are compared with discrete and previous non-splatting continuous plots with respect to rendering performance and visual quality. First, we compare results of all of the three rendering techniques with respect to their visual appearance. Then, some results obtained using resampling and progressive refinement are shown. We also investigate the relation of splat size and sampling resolution with respect to rendering performance and image coarseness. All measurements (including images) were produced with an implementation based on C++ and GLSL. The implementation was tested on a Windows PC with Intel(R) Core(TM) 2 Quad CPU running at 2.4 GHz and an NVIDIA GeForce 8800 GTX graphics card.

While discrete and prior continuous plots need not to be parameterized, our splatting algorithm depends on the choice of the kernel size and the number of samples. Figure 3 compares a discrete scatterplot of the "bucky ball" dataset (a common test-dataset in volume rendering representing a spherical fullerene) comprising $32 \times 32 \times 32$ data samples given on a uniform grid with the continuous versions rendered using our splatting approach and the projected tetrahedron algorithm [BW08]. The data dimensions are the original scalar value (horizontal axis) and its gradient magnitude (vertical axis). The arcs emerging in these types of plots can be used as an indicator for material boundaries. As the "bucky ball" dataset consists of several spheres (representing atoms in a fullerene molecule), material boundaries correspond to the sphere surfaces. See [KKH02] for further examples.

As discussed in Section 4, increasing splat size allows us to reduce the sampling resolution, while the level of coarseness of the resulting image is approximately maintained. The relationship between these parameters is illustrated in Figure 4. Here, every plot was rendered without progressive refinement but with different sampling resolutions and reconstruction kernel sizes. As expected, the images become smoother with increasing number of samples and with increasing splat size. Furthermore, coarseness is approximately maintained when doubling the number of samples with half of the kernel size and vice versa.

Figure 2 shows a performance analysis for different splat sizes and sampling rates, compared with traditional discrete scatterplots and continuous scatterplots using the original projected tetrahedra algorithm. From the measurement data, a linear dependency between sampling rate, splat size, and rendering performance can be concluded (note the log_2 scale of the x axis). Although performance decreases with the number of splats, the progressive refinement algorithm introduced in the previous section can still be used to achieve interactive framerates, as the total number of splats is divided into successive rendering frames. As a consequence, the algorithm scales well with dataset size and can easily be adopted for streaming data, as only a fixed number of samples is required for the individual rendering steps. This also includes large simulation data or time-dependent data, where visualization may be required in real-time. Finally, efficient rendering of statistical plots naturally facilitates the implementation of stacked displays or small multiples, where many instances of the same plotting technique are used to visualize different parts, projections, or subareas of the dataset. For example, the scatterplot matrix may be extended using progressive refinement.

Using the derivation in Section 3.3, the progressive refine-

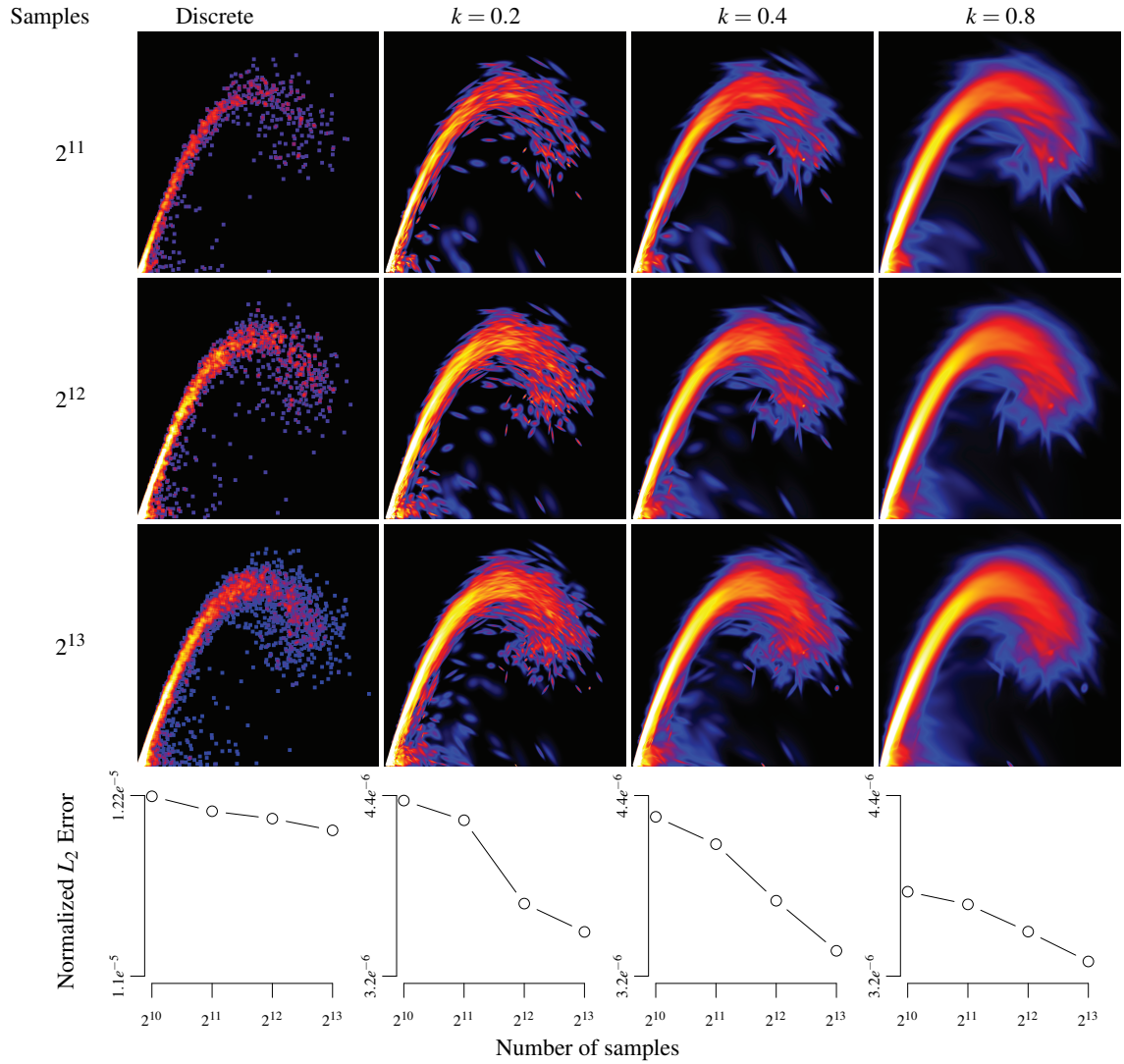

Figure 4: *The effect of the sampling resolution and the kernel size parameter with respect to the overall coarseness of the plot. In the matrix shown above, the number of samples increases from top to bottom while the splat size increases from left to right. In both cases, the corresponding images become smoother. At the same time, however, increasing splat sizes result in less accurate plots, as can be seen in the rightmost column. Here, the blur introduced by larger splats makes the image appear "wider". The bottom row shows the L_2 error of densities from the splatted plots with respect to a traditional, discrete scatterplot rendered with 10^7 samples of point-size four. While the convergence behavior is similar for all columns, the total error decreases with increasing smoothing factor k.*

ment algorithm can also be applied to parallel coordinates without change. The relation of splat size and sampling resolution regarding the coarseness of the resulting image remains the same, although smooth images are obtained with fewer samples due to the inherently stronger overdrawing of primitives in parallel coordinates. Figure 5 illustrates an example of a splatted continuous parallel-coordinates plot of a single timestep of the IEEE Visualization 2004 contest dataset "Hurricane Isabel" with a spatial resolution of $500 \times 500 \times 100$. The splatted image presented in Figure 5 was created using a subset of 10^3 samples. For comparison, a discrete version with 2.5×10^4 samples is shown. As with scatterplots, the splatted parallel-coordinates plot yields a good approximation to the continuous version with only a fraction of the samples given on the input grid.

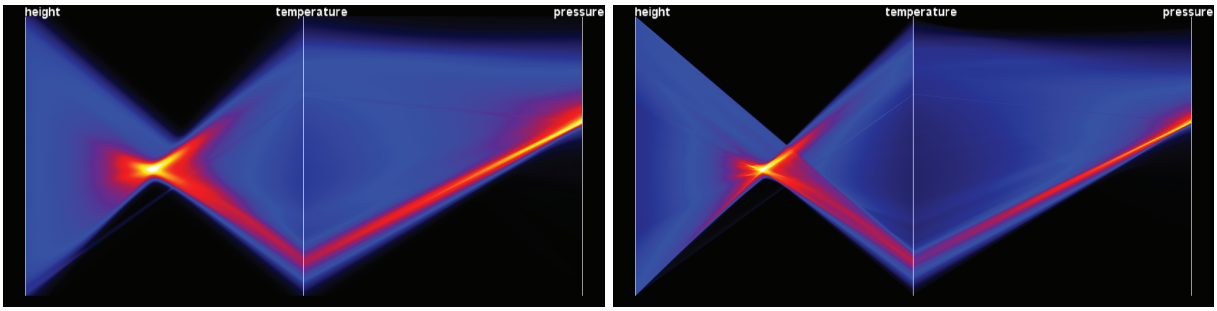

Figure 5: *Splatted parallel coordinates (10^3 samples, left) and discrete parallel coordinates (2.5×10^4 samples, right) showing three dimensions ("height", "temperature" and "pressure") of the first timestep of the simulation dataset "Hurricane Isabel". The plots indicate a negative correlation between "height" and "temperature" and a high density of samples with low "temperature" and mid-level "pressure". Note that the density distribution of the discrete version can be approximated using splatting with only a fraction of the original samples.*

6. Conclusion

We have presented a splatting algorithm for constructing continuous 2-D scatterplots and multivariate parallel-coordinates plots. The core element of our approach is the analytic transformation of a 3-D Gaussian kernel from the spatial domain to the scatterplot and parallel-coordinates domains, respectively. Discretized versions of these splats are pre-computed and stored in textures. During runtime, our progressive rendering algorithm subsequently adds more and more splats with decreasing variance to the plot, improving image quality. The main advantage of progressive rendering is that previews of the exact plot are available extremely fast, ideally supporting interactive data analysis especially for large data sets. Another advantage of splatting is that image quality can be gradually balanced with computation speed by adjusting the number of splats and splat size.

We have restricted ourselves to Gaussian kernels in the spatial domain. Our rendering approach is not necessarily limited to this kernel type. Other kernels would just require a different pre-computation of footprints. However, there is no apparent need for other kernels because Gaussian kernels with freely chosen width are capable of capturing density functions very well. An important open question for future work, though, is how we can determine the minimum number of splats beforehand that would guarantee a certain plot quality. In other words, can we develop the analog of Shannon's sampling theorem for sampling continuous scatterplots or parallel-coordinates plots?

Acknowledgments

In part, this work was supported by Deutsche Forschungsgemeinschaft (DFG) within SFB 716 / D.5.

Appendix

This appendix provides the derivation of Eq. (2). We first note that the density of a point ξ in the data domain is computed using the integration in Eq. (2). Without loss of generality, we assume that the spatial domain is unit-less.

As ξ_1 and ξ_2 both represent isosurfaces in the spatial domain that are planes, their intersection $\tau^{-1}(\xi)$ is a straight line that we write in parametric form as

$$\mathbf{l}(t) = \mathbf{p}' + t\mathbf{r}$$

with normalized tangent vector $\mathbf{r} = \frac{\nabla\xi_1 \times \nabla\xi_2}{\|\nabla\xi_1 \times \nabla\xi_2\|}$ and with

$$\mathbf{p}' = \mathbf{x}_0 + D_\tau^{-1}(\xi - \xi_0) \tag{7}$$

where D_τ denotes the Jacobian of τ evaluated at $\mathbf{x}_0$ and $D_\tau^{-1} = D_\tau^T (D_\tau D_\tau^T)^{-1}$ is the right inverse of D_τ.

Please note that Eq. (7) also defines a plane spanned by the gradients $\nabla\xi_1$ and $\nabla\xi_2$ containing $\mathbf{x}_0$. Now, solving Eq. (2) remains a matter of integrating a Gaussian over a line, which can be solved using a similar derivation as presented in [KPI*03]. We set $\mathbf{p} = D_\tau^{-1}(\xi - \xi_0)$ and note that $\mathbf{r}$ is perpendicular to $\mathbf{p}$ such that $\mathbf{r}^T\mathbf{p} = 0$. Also, $\xi_0 = \tau(\mathbf{x}_0)$. Using the derivation given in [KPI*03] (replacing K with D_τ^{-1}, $\mathbf{d}$ with $\mathbf{r}$, and $\mathbf{v}'_1$ with $\mathbf{p}$), we obtain:

$$\begin{aligned}
\sigma(\xi) &= \frac{1}{\|\nabla\xi_1 \times \nabla\xi_2\|} \int_{\tau^{-1}(\xi)} e^{-\frac{1}{k^2}(\mathbf{x}-\mathbf{x_0})^T(\mathbf{x}-\mathbf{x_0})} \mathrm{d}^2x \\
&= \frac{1}{\|\nabla\xi_1 \times \nabla\xi_2\|} \int_{-\infty}^{\infty} e^{-\frac{1}{k^2}(\mathbf{p}'+t\mathbf{r}-\mathbf{x_0})^T(\mathbf{p}'+t\mathbf{r}-\mathbf{x_0})} \mathrm{d}t \\
&= \frac{\sqrt{\pi}}{\|\nabla\xi_1 \times \nabla\xi_2\|} e^{-\frac{1}{k^2}\mathbf{p}^T\mathbf{p}} \\
&= \frac{\sqrt{\pi}}{\|\nabla\xi_1 \times \nabla\xi_2\|} e^{-(\xi-\xi_0)^T E(\xi-\xi_0)}
\end{aligned}$$

with $E = \frac{1}{k^2}\left(D_\tau D_\tau^T\right)^{-1}$.

References

[BA97] BOWMAN A. W., AZZALINI A.: *Applied Smoothing Techniques for Data Analysis*. Oxford University Press, Oxford, UK, 1997. 2

[BW08] BACHTHALER S., WEISKOPF D.: Continuous scatterplots. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1428–1435. 1, 2, 3, 4, 6, 7

[BW09] BACHTHALER S., WEISKOPF D.: Efficient and adaptive rendering of 2-D continuous scatterplots. *Computer Graphics Forum 28*, 3 (2009), 743–750. 2, 6

[CBB06] CARR H., BRIAN D., BRIAN D.: On histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics 12*, 5 (2006), 1259–1266. 1, 2

[CHU08] CHEN C., HÄRDLE W., UNWIN A. (Eds.): *Handbook of Data Visualization*. Springer-Verlag, Berlin, 2008. 2

[DGH03] DOLEISCH H., GASSER M., HAUSER H.: Interactive feature specification for focus+context visualization of complex simulation data. In *Eurographics / IEEE TCVG Symposium on Visualization* (2003), pp. 239–248. 2

[FKLT10] FENG D., KWOCK L., LEE Y., TAYLOR R. M.: Matching visual saliency to confidence in plots of uncertain data. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 980–989. 3

[HW09] HEINRICH J., WEISKOPF D.: Continuous parallel coordinates. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1531–1538. 1, 2, 3, 5, 6

[Ins85] INSELBERG A.: The plane with parallel coordinates. *The Visual Computer 1*, 4 (1985), 69–91. 2

[Ins09] INSELBERG A.: *Parallel Coordinates: Visual Multidimensional Geometry and Its Applications*. Springer, New York, 2009. 2

[KKH02] KNISS J., KINDLMANN G., HANSEN C.: Multidimensional transfer functions for interactive volume rendering. *IEEE Transactions on Visualization and Computer Graphics 8*, 3 (2002), 270–285. 2, 7

[KPI*03] KNISS J., PREMOZE S., IKITS M., LEFOHN A., HANSEN C., PRAUN E.: Gaussian transfer functions for multifield volume visualization. In *Proceedings of IEEE Visualization* (2003), pp. 497–504. 9

[LH91] LAUR D., HANRAHAN P.: Hierarchical splatting: A progressive refinement algorithm for volume rendering. *Computer Graphics 25*, 4 (1991), 285–288. 3

[LT10] LEHMANN D. J., THEISEL H.: Discontinuities in continuous scatter plots. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1291–1300. 2

[MW91] MILLER J. J., WEGMAN E. J.: *Construction of Line Densities for Parallel Coordinate Plots*. Springer, New York, NY, USA, 1991, pp. 107–123. 2

[MWCE09] MACIEJEWSKI R., WOO I., CHEN W., EBERT D.: Structuring feature space: A non-parametric method for volumetric transfer function generation. *IEEE Transactions on Visualization and Computer Graphics 15*, 6 (2009), 1473–1480. 2

[Nie92] NIEDERREITER H.: *Random Number Generation and Quasi-Monte Carlo Methods*. SIAM (Society for Industrial and Applied Mathematics), Philadelphia, 1992. 6

[NN11] NAGARAJ S., NATARAJAN V.: Relation-aware isosurface extraction in multifield data. *IEEE Transactions on Visualization and Computer Graphics 17*, 2 (2011), 182–191. 2

[PBM05] PRADHAN K., BARTZ D., MUELLER K.: *SignatureSpace: A multidimensional, exploratory approach for the analysis of volume data*. Tech. rep., Eberhard Karls Universität Tübingen, 2005. 2

[PD84] PORTER T., DUFF T.: Compositing digital images. *Computer Graphics (Proceedings of SIGGRAPH 84) 18*, 3 (1984), 253–259. 6

[SSD*08] SCHEIDEGGER C. E., SCHREINER J. M., DUFFY B., CARR H., SILVA C. T.: Revisiting histograms and isosurface statistics. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (2008), 1659–1666. 1, 2

[TPM05] TORY M., POTTS S., MÖLLER T.: A parallel coordinates style interface for exploratory volume visualization. *IEEE Transactions on Visualization and Computer Graphics 11*, 1 (2005), 71–80. 2

[Tuk77] TUKEY J. W.: *Exploratory Data Analysis*. Addison-Wesley, 1977. 2

[Wes89] WESTOVER L.: Interactive volume rendering. In *Proceedings of the 1989 Chapel Hill workshop on Volume visualization* (1989), ACM, pp. 9–16. 2, 3

[Wes90] WESTOVER L.: Footprint evaluation for volume rendering. *Computer Graphics (Proceedings of SIGGRAPH 90) 24* (1990), 367–376. 5

[WL97] WEGMAN E. J., LUO Q.: High dimensional clustering using parallel coordinates and the grand tour. *Computing Science and Statistics 28* (1997), 361–368. 2

[XLS10] XU L., LEE T.-Y., SHEN H.-W.: An information-theoretic framework for flow visualization. *IEEE Transactions on Visualization and Computer Graphics 16*, 6 (2010), 1216–1224. 2

[ZCQ*09] ZHOU H., CUI W., QU H., WU Y., YUAN X., ZHUO Z.: Splatting the lines in parallel coordinates. *Computer Graphics Forum 28*, 3 (2009), 759–766. 2

[ZK10] ZHAO X., KAUFMAN A.: Multi-dimensional reduction and transfer function design using parallel coordinates. In *IEEE/EG International Symposium on Volume Graphics* (2010), pp. 69–76. 2