Splatterplots: Overcoming Overdraw in Scatter Plots

Adrian Mayorga, *Student Member, IEEE,* and Michael Gleicher, *Member, IEEE*

Abstract—We introduce Splatterplots, a novel presentation of scattered data that enables visualizations that scale beyond standard scatter plots. Traditional scatter plots suffer from overdraw (overlapping glyphs) as the number of points per unit area increases. Overdraw obscures outliers, hides data distributions, and makes the relationship among subgroups of the data difficult to discern. To address these issues, Splatterplots abstract away information such that the density of data shown in any unit of screen space is bounded, while allowing continuous zoom to reveal abstracted details. Abstraction automatically groups dense data points into contours and samples remaining points. We combine techniques for abstraction with with perceptually based color blending to reveal the relationship between data subgroups. The resulting visualizations represent the dense regions of each subgroup of the dataset as smooth closed shapes and show representative outliers explicitly. We present techniques that leverage the GPU for Splatterplot computation and rendering, enabling interaction with massive data sets. We show how splatterplots can be an effective alternative to traditional methods of displaying scatter data communicating data trends, outliers, and data set relationships much like traditional scatter plots, but scaling to data sets of higher density and up to millions of points on the screen.

Index Terms—Scalability issues, visual design, perception theory, statistical graphics.

1 Introduction

Scatter plots are a simple, intuitive and natural way of visualizing two dimensional point data. Scatter plots can display data trends and correlations between any two dimensions. They can make outliers easy to identify because regions with higher density of points will be grouped perceptually. Additionally, scatter plots offer a means for comparing different data sets when plotted on the same axes. These properties make scatter plots good for exploring data sets and communicating interesting findings. Unfortunately, scatter plots become less effective as the overlap within points increases.

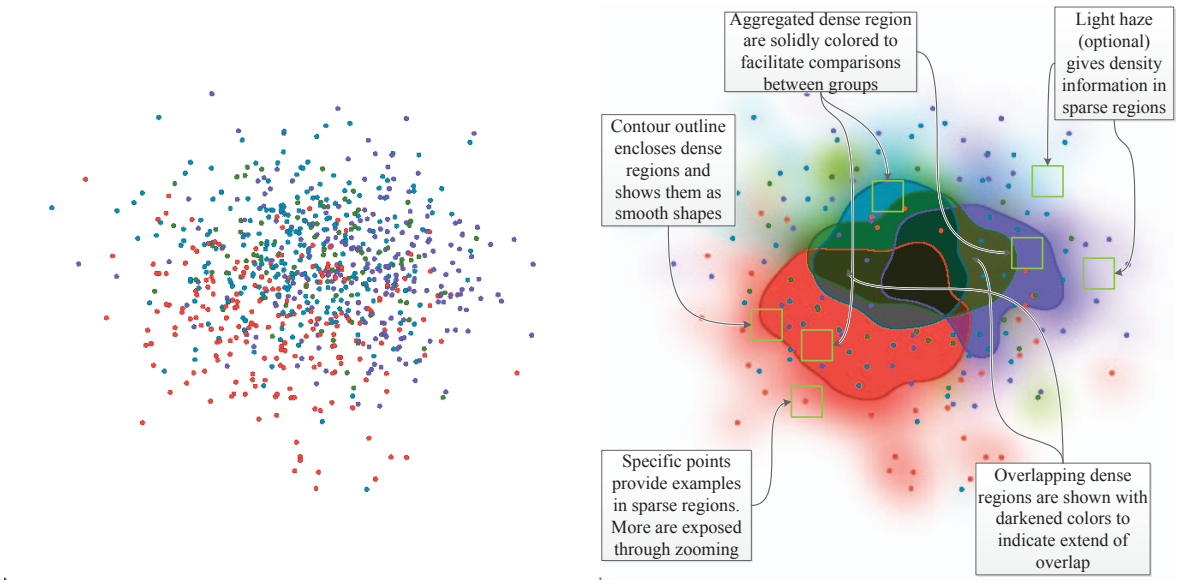
Overdrawing occurs when the glyphs that are used to visualize data points overlap. This overlap can become so severe that it is impossible to perceive the number of points in a given region of the scatter plot. This interferes with the viewer's ability to group points perceptually and spot outliers. Overdrawing increases as the available drawing space per glyph decreases. High density regions, large numbers of points, and multiple data sets all contribute to the overdraw problem. Even if we decrease the size of the glyphs to be a single pixel, an increasing number of data sets contain many more points than there are pixels in a monitor. Additionally, many analysis and visualization techniques, such as scatter plot matrices (SPLOMS), call for several different concurrent plots, further diminishing the amount of available screen space.

In this paper we introduce *Splatterplots*, a novel presentation of point data that addresses weaknesses in scatter plots to better scale to larger datasets. Our key idea is to limit the amount of visual complexity in any screen space area. Splatterplots abstract point data so that the amount of information shown in any area of screen space is bounded. Dense regions are shown as smooth shapes, and outlying points are subsampled so they do not exceed a specified visual density. A screen space metric controls both the amount of detail in the smooth shapes and the display density of outlier glyphs, enforcing a limit on the amount of visual complexity. The details removed in a Splatterplot can be revealed through interaction mechanisms such as zooming and panning, enabling fast exploration of massive datasets. Continuous zooming automatically reveals abstracted details since abstractions are determined in screen space. Splatterplots allow a user to see trends and patterns in datasets even as the number of points grows large relative to the diagram size. Particularly, subgroup relationships and ranges can be clearly observed, even as the number of data subgroups grows past two or three.

This paper discusses the design of Splatterplots, describing how perceptual research led us to a design that uses abstraction and multiple visual representations. We describe a process for creating Splatterplots that incorporates kernel-density estimation and specially tuned color blending. We integrate zooming techniques that reveal abstracted information as it becomes feasible to perceive it. A GPU-accelerated

• *A. Mayorga and M. Gleicher are with the Department of Computer Sciences, University of Wisconsin, Madison, WI, 53706. E-mail: adrm@cs.wisc.edu,gleicher@cs.wisc.edu*

Digital Object Indentifier 10.1109/TVCG.2013.65



(a) scatter plot (b) splatterplot with explanations

Fig. 1. Same dataset shown as a standard scatter plot(a), and a Splatterplot(b). The callouts point out the features of our visualization. Dense regions are aggregated into smooth contours, remaining points are sampled to remove clutter while still showing some individual points, and density information in sparse areas is shown as density clouds.

implementation allows interaction, even with large data sets. We demonstrate the usefulness and flexibility of Splatterplots on examples of several hundred thousand scatter points.

Contributions: The central contribution of this paper is a new technique for displaying scattered data that remains interpretable as the number of points grows large. Our design is based on two key ideas: to explicitly perform abstraction to reduce visual clutter; and to perform these abstractions in screen space. To realize these ideas, we provide a design that uses different encodings for sparse and dense regions, each bounding information density in screen space. The design reduces the amount of information presented so that the main features of the data remain visible. Additionally, abstracting dense regions as smooth regions enables data subgroup comparisons. Our novel design is supported through a pipeline of steps that couple standard data abstraction tools with interaction techniques, all performed with the aid of the GPU, enabling our approach to scale well. Our results show useful depictions of massive datasets.

2 Related Work

The problem providing scalable views of scatter data has been approached in several ways. These prior efforts motivate our work.

Perhaps one of the most common ways of dealing with dense scatter data is the use of density estimations. Whether the method is parametric or not, these methods involve computing a smooth function over the area of interest. These densities can then be displayed as a continuous scalar field, for example by mapping the values to visual parameters or creating contour plots. Unfortunately, existing methods for scalar field display are problematic. Methods do not handle outliers well, as isolated points create small regions of low density. Methods for the denser regions do not scale to multiple data series. While effective encodings of a single value are well known, for example color ramps from ColorBrewer [1], encoding multiple, overlapping fields is challenging. Contour plots quantize the density values to better expose shape, but fail to capture outliers, become complex quickly with data details, and fail to scale to multiple data series. Hexbin plots and 2D histograms similarly quantize density in local regions, but dense regions created by such plots do not create perceptually smooth contours, making them hard to group and compare visually. Our approach builds on the ideas of density estimation, color mapping, and contour finding but adds explicit consideration of abstraction in screen space to limit visual complexity, and handles outliers specifically.

Alpha bending can be used to show each individual scatter plot point semi-transparently [2]. This common and simple form of density estimation can be implemented efficiently using using accumulation in the frame buffer. While it does not address any of the shortcomings of the more general approach, it does suggest an efficient implementation using graphics hardware, an approach we build upon.

The general challenge of visualizing multiple scalar fields has lead to techniques that are unsatisfying for

display density information from scatter data. Color blending does not offer a satisfactory solution for even two sets as the non-linearity of perception make a quantitatively interpretable blend difficult [3], [4]. Methods such as Texton Maps [5], Attribute Blocks [6] and Color Weaving [7] all trade spatial resolution for value resolution, and are primarily effective at indicating regions of constant value, not continuously varying quantities such as density. The methods also fail to scale in the number of fields. Our approach provides spatial resolution by explicitly showing contours, and limits the amount of density information conveyed so that we can employ ideas inspired by the multi-field techniques to provide for multiple overlapping data subgroups. The contours created by our technique are analogous to overlapping sets. Bubble Sets [8] and Simonetto et. al. [9] show relationships between overlapping sets by enclosing related groups with contours. While these techniques effectively show relationships between different sets, they fail to scale directly to visualizations representing large amounts of data points.

The dense point problem has also been addressed through the use of clustering, sampling, and filtering. Ellis and Dix [10], [11] use sampling as a means to reduce the density of points shown. Bertini and Santucci [12] model the underlaying data density to perform uniform and non-uniform sampling. Ellis and Dix [13] have also compiled a taxonomy of clutter reduction techniques used in information visualization. While these approaches can very effectively reduce the number of points in the dataset to a manageable number, the challenge of handling multiple sets remains. Our approach effectively performs clustering by grouping dense regions of points into continuous regions. These solutions provide a perceptually-motivated clustering that affords effective visual presentation, and adapts to interaction. We use sampling strategies in less dense regions.

Dang et al. [14] present a method for dealing with overlapping data by stacking points in the third dimension, but this method does not scale well with large datasets. Tory et al. [15] compare similar encodings, but similarly do not consider issues of scalability.

Generalized scatter plots [16] present a way of nonlinearly warping the scatter dimensions to avoid overdrawing. While this makes the shape of the data in the dense regions more apparent without losing outliers, it is difficult to see the relationship between the dense regions and the non dense regions. The warps are also hard to understand intuitively, but they can be visualized by overlaying Voronoi shapes over the points. This makes it hard to overlay different datasets on top of each other, since the warp must be computed for the aggregated set. Variable binned scatter plots [17] explicitly groups regions of space into subplots and highlight different statistical properties, such as median and internal quartiles. Again,

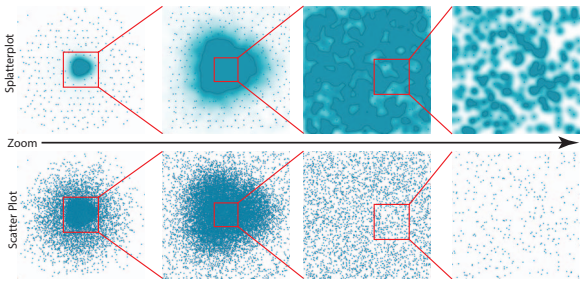


Fig. 2. Effects of zooming on detail in a splatterplot. Details emerge on the contour as zooming occurs, the size of the contour features remains constant. Similarly, zooming reveals more points, but the density of points remains the same. Scatter plots are shown in the bottom row as reference.

while this may work for one dataset, comparison with another is difficult. Additionally, the visualization is not as intuitive as scatter plots, and requires a significant amount of training to understand.

Scatter plots have been extended in several ways. For example, extensions support higher dimensional data, such as interactive SPLOMs [18]. Such techniques do not address the scalability issues in scatter plots, if anything they exacerbate it by creating more, smaller plots. This problem can be ameliorated by view selections techniques such as those presented Sips et. al. [19] and Wilkinson and Wills [20]. Other extensions that encode additional data variables onto the points, such as bubble plots or shaped glyphs, generally fail to scale to dense point sets as they also exacerbate the overdraw problem.

Several studies have examined specific aspects of the perception of scatter plots. These include the perception of correlation [21], discrimination of symbol size [22], symbol lightness [23], and symbol contrast [24]. While these do not explicitly deal with the perception issues caused by overdrawing, a full solution towards perceptually scalable scatter plots should take these findings into account.

Continuous scatter plots [25] solve an orthogonal problem. They are designed to generate 2D continuous histograms of values from a scalar or vector field with a spatial embedding, in an effort to bring the advantages of scatter plots to scalar and vector fields.

3 Design and Motivation

The amount of information that can be displayed and observed in a unit of screen area is bounded by perceptual limits and display resolution. In traditional scatter plots this manifests both as actual overdraw, and as visual clutter. Showing too many points creates visual clutter and makes judgments about the data harder, effectively causing perceptual data loss. The main goal of our design is to achieve perceptual scalability: as the number of points grows, the display

remains readable. Since data loss is inevitable, we chose to abstract information explicitly, rather than fall pray to shortcomings in visual perception and display density. Figure 2 shows the same dataset shown as a Splatterplot and as a scatter plot, with zoom level increasing from left to right. Splatterplots maintain a readable display, regardless of zoom level.

Our main intuition is that since the problems in scatter plots arise from limitations in the visual perceptual system and display density, the right space to think about abstractions is in visual perceptual space, and not data space. Visual perception may depend on many factors, such as the display, distance to display, the data being visualized, and even the viewer. Also, the only way to judge abstraction parameters is by visual inspection. This means that the system should be have a way to quickly and effectively visually explore the space of abstraction parameters.

This line of reasoning leads us to two main design principles. First, we perform data abstractions in screen space, which acts as a proxy for visual perceptual space. Second, our system must be able to handle large datasets at interactive rates to quickly explore abstraction parameters, enabling the user to chose good parameters for the task at hand. Consequently, these two principles, screen space abstraction and fast performance, result in a continuous zoom that smoothly reveals abstracted details. With these design principles in mind, we examine common methods for dealing with overdraw and clutter, including density based displays and point filtering, in order to create a visualization that combines the strengths of both while removing drawbacks of both.

3.1 Single Encodings

First we look at visualizations that use a single encoding for displaying point data. Specifically, we look at density based visualization and point filtering approaches, and examine how one might extend these approaches to meet our design goals.

3.1.1 Density

Density and contour plots are a common way of dealing with overdraw issues, since they remove the idea of discrete glyphs for data points. However, most implementations of density and contour plots implement rules of thumb or statistical models to compute the ideal amount of smoothing in data space. While these approaches have merits when making statistical queries about the data, they completely ignore how the resulting plots will be perceived by viewers.

A simple way to make density plots conform to our first principle (screen space abstraction) is to directly tie the amount of smoothing used to create density functions directly to screen space. However, we need to do this in such a way that we can generate visualizations at interactive rates. While density displays might do an adequate job of reducing clutter, too much information is lost in the sparse regions. Also, density displays do not show the original data points, which is important for doing direct queries of the data when the specific values of points of interest are important.

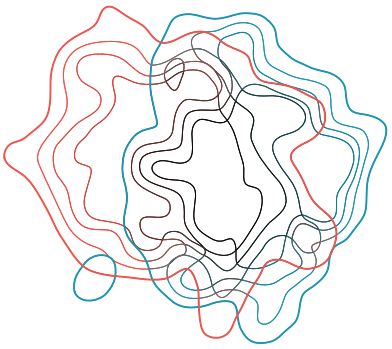


Fig. 3. Two datasets drawn with as a contour plot with concentric contours. Parallel contours make it difficult to perceive any single level. The intersecting rings create many small regions, adding considerable amounts of visual complexity that is difficult to untangle.

3.1.2 Point Filtering

Point filtering approaches address the overdraw problem while still showing a portion of the original data. While different approaches have different sampling methods for reducing the amount of visible points, one of the main considerations is whether to sample uniformly in data space, or sample such that the resulting figure has a constant density. Even if we tie filtering parameters to screen space this question still remains. Uniform data sampling retains some density information, but may still lead to visual clutter in datasets with large numbers of points, while constant density sampling bounds the amount of visual clutter but removes a lot of density information. In other words, point filtering approaches work well in the sparser regions of scatter plots, but not as well in the dense regions where overdraw occurs.

3.2 Dual Encoding

The previous section suggests that dense and sparse regions in scatter plots behave differently at a perceptual level, and therefore should be treated differently. We combine elements of point filtered scatter plots, density displays, and contour plots to create a visualization based in a dual encoding.

The Gestalt law of proximity states that perception tends to group similar objects that are close together as part of a greater whole [26]. Because of overdraw, visual estimates of density within the group may be imprecise or impossible, since the shape of the perceived grouping may not match the distribution of

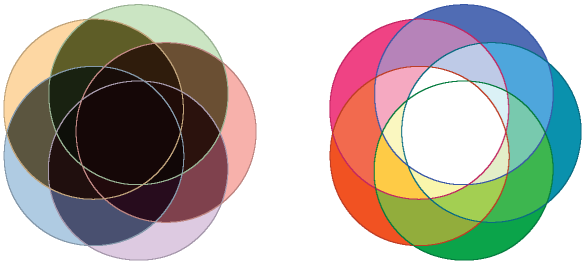


Fig. 4. Example color sets from our system. These figures show how hue is used to denote group, while lightness and saturation is used to encode amount of overlap. On the left, a set of pastel starting colors is used, on the right, increasing overlap is shown with increasing lightness.

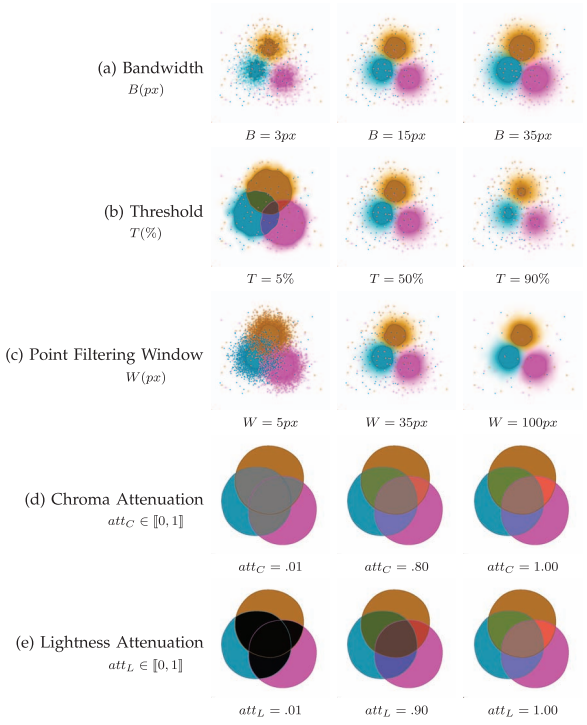


Fig. 5. Visual representation of all the user parameters in Splatterplots. The middle column are our default settings, which work well in most settings.

density in the underlying data. Instead, we use contours to aggregate points, which assists in perceptual grouping and explicitly shows the shape of dense data regions.

The literature on contour perception suggests several specific perceptual goals for making contours as easy to perceive as possible. First, concentric contours such as those found in contour plots should be avoided as they create parallel flankers, making it difficult to perceptually integrate any specific contour [27] (see Figure 3). Second, contours should be locally linear [28] and globally circular or elliptical [29]. Therefore, our design represents regions as smooth contours and avoids concentric rings.

Dense regions where extensive overdraw obscures the true shape of the data are represented as smooth contours with a solid fill, while the sparse region is encoded with a mix of density data and filtered points. This encoding allows the user to clearly perceive the shape, location, and extent of the densest regions of data, while still being able to get an idea about the rest of the data. Additionally, aggregating dense regions of data as a single shape frees up significant portions of the color ramp, allowing for increased detail density in the sparse region. This representation works well when displaying plots where all the data points belong to the same group. When multiple groups are present, the display needs to support differentiation among the groups, while introducing minimal amounts of visual clutter. Our approach uses color for this purpose.

3.3 Color Blending

Our approach uses color to encode multiple things (group membership, density, and amount of overlap with other groups), which means that we need to be careful about how these properties map to color. Color has several related components separated into three different groups: lightness/brightness, saturation/colorfulness/chromaticity, and hue. Generally three, one from each group, are needed to describe color. Lightness, chromaticity, and hue are a possible combination of parameters that are generally considered to be orthogonal [30].

Our approach uses hue to differentiate between data subgroups. This leaves both chromaticity and lightness open for encoding density information for each group, as well amount of overlap between groups. For encoding density in sparse areas, we essentially interpolate from white for zero density to the group color for max density. For amount of overlap, our approach attenuates the blended color between groups, pushing the resulting color towards black. This type of color blending strategy effectively ties blackness to amount of overlap, albeit at the expense of clearly encoding which groups belong to the overlapping region, as illustrated in Figure 4. However, this blending allows us to clearly show group relationships effectively with a small number of groups when using the dual encoding, while still showing points and density information in the sparse regions (see §4.3 and §4.6 for details).

3.4 Splatterplots

Splatterplots make use of the dual encoding and color blending to create easy to perceive displays of dense data with multiple subgroups, in an interactive system that has few parameters. Figure 5 shows a visual summary of how these parameters behave. These

parameters can be changed with a simple slider, and the system provides the user with immediate visual feedback. In our uses, we found that bandwidth, threshold level, and the size of the point filtering window were the most used parameters.

4 SPLATTERPLOTS

The design described in Section 3 is implemented by the multi-representation pipeline outlined in Figure 6. The process treats each data subgroup separately to produce dense regions and sparse points, and then combines these into a single splatterplot. For each group, a density function is computed (§4.1), and a thresholded distance function is used to identify the contour (§4.2). Unaggregated subgroup data points are then sampled to compute the visible outliers for the current view (§4.5). Colors are assigned to each subgroup (§4.3) in a manner that affords perceptual blending (§4.4) when the regions from the different subgroups are combined. With the help of the GPU, these steps are computed fast enough to allow for interactive exploration, including zooming, panning, and changing the value of screen space parameters. Screen space parameters control the amount of smoothing in the density computation and the sampling density of points in the sparse region, bounding the visual complexity of each group in terms of screen space. We empirically test the limits of our approach (§4.6), including an analysis of performance, color blending, and visual clutter.

4.1 Kernel Density Estimation

Point aggregation into contours is achieved through density estimation and thresholding. As a first step we compute a density scalar field from the data points of each subgroup. Kernel Density Estimation (KDE) provides a well-studied, statistically sound, family of methods that is used in other similar contexts [31], [32]. Because our points are dense and our samples are close together and regularly spaced, we can use a simple version that allows us to leverage the GPU for efficient computation.

KDE estimates continuous density values by summing the contribution of discrete samples around a point based on a kernel function. Formally, for n points $\mathbf{x}_i \in \mathbf{X}$ where $\mathbf{X}$ is the set of samples, the density estimate $\hat{f}$ at point $\mathbf{q}$ can be computed by

$$\hat{f}(\mathbf{q}) = \frac{1}{n}\sum_{i=1}^{n} K_B(r_i)$$

where $K_B(x) = \frac{1}{B}K(\frac{x}{B})$, K is the kernel function B (Figure 5a) is the bandwidth parameter, and $r_i = \|\mathbf{q} - \mathbf{x}_i\|$. B controls the amount of smoothing. We use K to equal a Gaussian function

$$K(r) = \frac{1}{\sqrt{2\pi}} e^{\frac{-r^2}{2\sigma^2}}$$

and therefore the B parameter effectively becomes the variance of the Gaussian kernel. Our implementation accumulates the points into a frame buffer texture creating a 2D scalar field of point counts per pixel. This scale field is convolved with the kernel K, blurring the field using a two pass approach (one horizontal, one vertical), exploiting the separability of the 2D Gaussian kernel. Any noticeable artifacts caused by accumulating points at the pixel level are removed by blurring. While the choice of kernel function $K(r)$ is important, the bandwidth parameter B has the biggest influence on the shape of the density field. In our technique, B is equal to the current value of the screen space abstraction metric.

4.2 Distance Transform

In order to ensure that outlier points do not clutter contours we filter out any points that lie too close to the contours (Figure 5c). Instead of calculating the distance to the contour for each data point, we use a distance transform that calculates the distance to the closest point on the contour at each point in space. This creates a scalar field that we can efficiently sample. A distance transform also facilitates the anti-aliased rendering of contour outlines in a shader by allowing each pixel to query whether it falls in the immediate vicinity of the contour. The boundary of the contour is determined by the relative threshold parameter T (Figure 5b).

To compute the distance transform our implementation uses a vector propagation algorithm tuned to the GPU, known as the Jump Flooding Algorithm (JFA) [33], which is initialized by thresholding the previously computed density field.

4.3 Coloring and Contouring

The final step in contour aggregation is to represent the dense regions as an enclosed smooth shape. This step takes as input the density and distance fields, maximum density value, relative thresholding level T (Figure 5b), foreground and background color. The output is a fully colored and outlined contour that represents the aggregated dense points of a subgroup.

Foreground colors for each data subgroup will be used in the later blending stage and require special considerations. We work in the CIE Lab and LCH color spaces for most of our color computations. CIE Lab is a perceptually motivated color space where euclidian distances can be used to approximate perceptual distances [30]. Color choices need to be easily distinguishable and must work well with our blending strategy, which we use to deal with multiple data subgroups. Our blending strategy uses the lightness channel in later stages, disallowing the use of precomputed color schemes such as those in ColorBrewer [1]. Therefore, our approach computes colors to be isoluminant and as far away in Lab space as possible

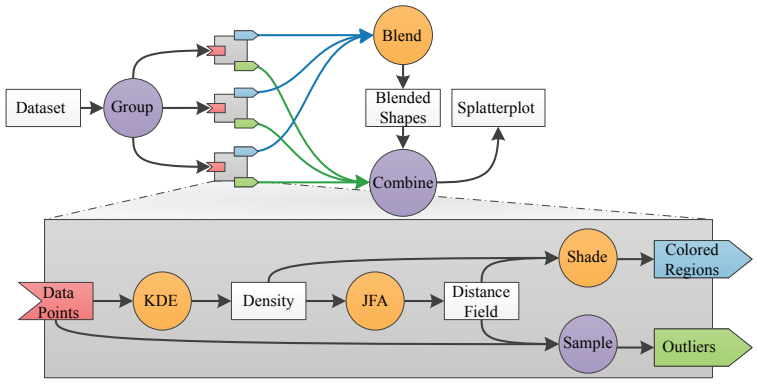


Fig. 6. Splatterplot pipeline. Rectangles are data representations, while circles are operations. Orange operations are performed in the GPU through shaders, while purple ones occur in the cpu.

Fig. 7. Comparison of different blending paradigms. Notice how in RGB blending, blue hues tend to dominate, causing blended regions to appear to belong more to one set than another.

by choosing an L value of 74.5 and evenly spacing the colors along hue angles H. We allow the computed colors to be adjusted manually, however that has not been done for any of the examples in this paper, except for Figure 4. We use a background color of white.

Using the density field, any pixel value that falls at or above the specified relative threshold level is assigned the foreground color. We draw a three pixel outline around the isocontour assigned by the relative thresholding level. This is largely facilitated by the distance field created in the previous step.

We show density information in sparse regions by using the density values to modulate the lightness and saturation of each pixel towards white. We also take precautions so that during the blending step the colors in the sparse region are not blended with those in the dense regions of other groups.

4.4 Blending and Combining

Once the aggregated contours and sampled outliers have been computed for each subgroup we combine all of the results to generate the Splatterplot.

In general, color blending is not an effective means to indicate set membership between overlapping sets, especially for more than three sets. Although we use color blending to provide limited set membership information, we emphasize the amount of overlapping sets in each region using lightness and chroma parameters. This provides a coarse measure of overall density while distinguishing regions with different amounts of overlap.

Color blending is computed as straight forward weighted averaging in LAB space, with one caveat. We attenuate both chroma (saturation) and lightness (brightness) by a factor based on the total density weight at the current pixel. For each group, a pixel that falls inside the aggregated region is assigned a weight of 1, while those outside are assigned a weight based on the local density. This means that as the total density weight per pixel increases, the resulting color gets closer to black. The final blend of n colors c_i each with density weight w_i can be computed by the formulas:

$$
\begin{aligned}
w &= \left(\frac{1}{n}\sum_{i}^{n} w_i\right) \\
(L,C,H) &= LABtoLCH\left(\frac{1}{w}\sum_{i}^{n} c_i w_i\right) \\
color_{LAB} &= LCHtoLAB\left((att_L)^{w-1}L, (att_C)^{w-1}C, H\right)
\end{aligned}
$$

where att_L, att_C (Figure 5d,e) are the lightness and chroma attenuation factors respectively, and $LABtoLCH$ and $LCHtoLAB$ are conversions between the two color spaces. Attenuation factors are controlled by the user interactively. Figure 7 shows the difference between RGB, LAB, and blending with attenuation. Figure 4 shows how this coloring scheme cleanly separates regions that have different number of member regions, while at the same time making it much harder for the viewer to misjudge the members of a particular region.

4.5 Sampling Sparse Regions

We want to retain any points not aggregated by contours without creating visual clutter or overdraw. To do this, our approach imposes a screen space limit

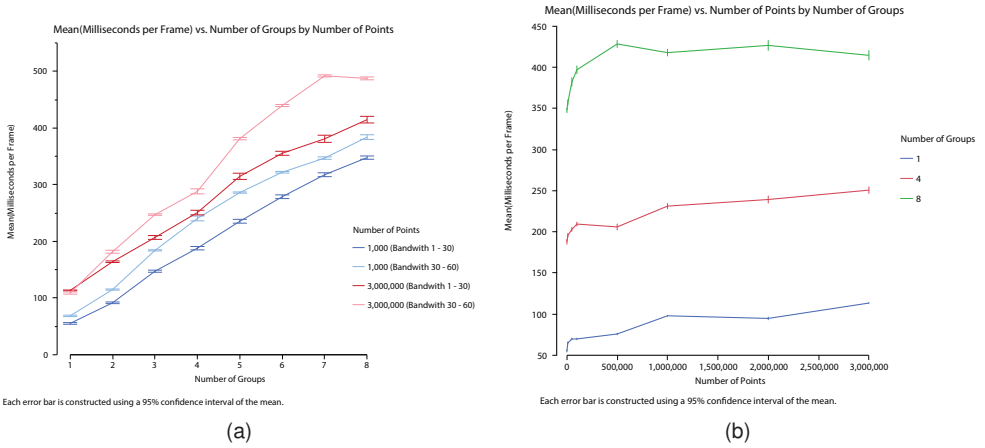


Fig. 8. Two different views of some of our performance data. Figure 8a shows how performance is linear with respect to the number of groups. Figure 8b shows our performance data, averaging values with bandwidth from 1 to 30, with the number of points in the horizontal axis. This graphs shows consistent performance when the number of groups is held constant, even as the number of points grows to 3 million.

on how close points can be to each other, as well as how close points are to contours.

Once the contours have been generated the outliers need to be identified. Ideally, we would like this procedure to scale linearly with the number of data points. Our implementation does this by first reading back the distance field created during the JFA step into program memory. Our approach then performs a simple nearest neighbor query on the distance field for each point. Only points at a distance greater than the screen space metric value proceed to the sampling stage.

For the sampling stage, our approach divides the current viewing area into a grid where each bin has a size determined by screen space parameter W (Figure 5c). Then, we simply collect the allowed points into the bins and render the first point in each bin. This simple heuristic works well in practice, but begins to slow down considerably as the number of points grows beyond 100,000. In order to enable interactive exploration of massive datasets, we perform a pre-filtering step with the help of the GPU.

To do the pre-filtering, we render all of the points into a texture. The Z value of each point is set to be a normalized index. We do the same for the color of each point. During rendering, we enable depth testing, and set the depth testing function to pass Z values that are less than or equal to the previous value. After rendering, we read back the texture information, and recover an index for each nonzero value by multiplying by the total number of points. This procedure essentially pre filters all of the points, setting an upper bound of $width * height$ where these are the height and width of the current viewport.

This simple heuristic is fast, effectively randomly choosing a point per bin to convey that there are data points in the region, and sets a minimum on the average distance between points that creates a screen space density limit on the shown outlier points.

4.6 Implementation

We performed several experiments to analyze the performance and behavior of several of our design decisions and our prototype implementation. Our prototype system was written as a 64-bit .Net application in C#, and uses OpenGL through the OpenTK library. Our performance analysis was conducted in a machine with an Intel i7-3770k processor, 16GB of RAM, and an NVIDIA GTX 670 graphics card, all under Windows 7 64-bit.

4.6.1 Performance Analysis

One of our main design goals was to make a system that ran at interactive rates, even when dealing with massive datasets. To test the behavior of our system under heavy stress we ran a series of performance experiments. We first created several synthetic datasets with Gaussian distributions. These range from having 1000 points to $3,000,000$ points. Along with varying the number of points, we also split up the datasets into groups, varying from one to eight different groups. For each dataset, we then created a splatterplot in a viewport of $700 * 700$ pixels, and varied the smoothing parameter. For each possible combination of these variables, 100 frames were rendered, measuring the amount of time each frame took to render.

After collecting all of the data, we averaged the frame rates and performed a simple analysis. Figure 8 shows a graphical summary of our results. From the graphs, we can see that the dominating factor affecting performance is the number of groups in the splatterplot. This is to be expected: the most time consuming steps in the process, such as computing

# of Groups	# of Colors	att_C	att_L	Min Dist
2	3	0.32	0.43	82.03
3	7	1.00	0.84	31.51
4	15	1.00	0.88	19.40
5	31	1.00	0.95	10.13
6	63	1.00	0.98	1.30
7	127	1.00	0.96	2.49
8	255	1.00	0.98	0.45

Fig. 9. Optimal att_C and att_L parameters for our blending strategy. These combinations of parameters maximize the distance between possible resulting colors.

the distance transformation and density estimation, must be performed independently for each group. We can also see that in the most extreme scenario (3 million points, over 30px bandwidth, and 8 different groups) we still achieved a frame rate of about 2fps. Realistically, bandwidth parameters over 30px are not very useful as they tend to over smooth the data. In a more realistic data scenario of 3 million, about 15px bandwidth, and about 4 different groups, we would observe a frame rate between 4-5fps.

4.6.2 Color Blending Analysis

We designed our color blending to scale beyond 3 different groups by blending colors in CIELAB and attenuating Lightness and Chroma. We also tested the limits of our color blending approach to verify that our design goals were met. For numbers of groups G varying from 2 to 8, we computed initial colors as we describe in Section 4.3. We also calculated the resulting color from each possible combination of initial colors, using our blending strategy from Section 4.4, varying the Chroma and Lightness (att_C and att_L respectively) attenuation parameters from 0 to 1 in increments of .01.

This results in a list of colors, for which we measured the pairwise Euclidian distance in CIELAB space. For each G we then found the att_C and att_L parameters that maximized the minimum distance between different colors.

In CIELAB space a Euclidian distance of over about 2.5 is said to be the just noticeable difference [34]. In other words, as long as two colors are further than 2.5 units in lab space, they should be perceptually different. From the table, we can see that for up to 5 groups, our color blending strategy can achieve a perceptual distance between colors well over this threshold. At 6 groups, main groups are still distinct, and in overlapping regions the amount of darkness still conveys amount of overlap.

4.6.3 Visual Clutter Analysis

A key idea of splatterplots is that they bound the perceptual density of the results: as the amount of data increases, the complexity of the display remains constant. To quantify complexity, we can use the metric of visual clutter as a proxy for the more subjective and complex concept. Rosenholtz et al. [35] provide an image-based clutter metric and show its utility for computer displays [36]. To assess how our methods scale perceptually, we performed an experiment using the publicly available implementation of Rosenholtz's [35] visual clutter metric. To test the clutter we used several synthetic datasets with gaussian distributions, varying from 50,000 to 3,000,000 points. The number of groups in each data set also varied from one group to eight. For each dataset, we started with a viewport that includes all of the points in the dataset, and then slowly zoomed in to the center. At each zoom step, we saved images that correspond to a regular scatter plot and a splatterplot. All of the resulting images where then run through Rosenholtz's visual clutter metric.

Figure 10 show a graphical summary of the results. The results show that the visual clutter for splatterplots remains fairly constant, while the clutter for scatter plots is higher and depends heavily on the number of points in the viewport. The trend line for the scatter plot clutter behaves as expected. When there are very few points, overdraw is minimal, and therefore clutter. However, as points begin to become closer, some overdraw begins to occur, and the space between points becomes about the same size as the points themselves. As overdraw begins to increase dramatically, the points become a blob with a simple texture, and therefore the visual clutter drop, but at the same time the plot no longer shows an accurate representation of the data.

5 Case Studies

Splatterplots are useful in a variety of applications including multidimensional data exploration and spatially embedded event visualization (see Figure 13). Here we describe scenarios on standard model datasets, along with some real datasets.

5.1 Synthetic Data Sets

To illustrate splatterplots, we provide some examples on simple data sets. While the benefits of splatterplots are most clear in multi-group comparison problems, where there are few good alternate approaches, they are also effective in single class problems. Figure 11 shows the application of our tool to the well-known "pollen" data set, allowing the dense region to be identified, and the hidden word to be revealed through zooming. Figure 12 illustrates several methods for large scatter data set visualization on a synthetic example consisting of 8 groups of points each sampled from a Gaussian distribution. The splatterplot technique is able to convey the circular shape of the core of each gaussian, provide a sense of the overlaps between regions, and indicate that there are

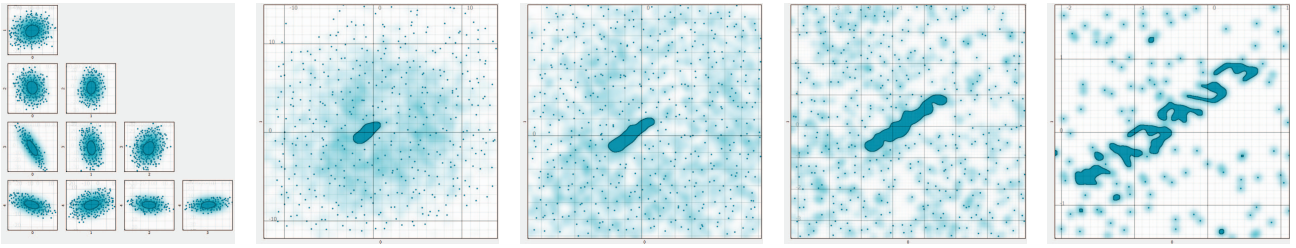


Fig. 11. Applying splatterplots to the pollen data set. Dense regions are notable in large-scale views, and details are exposed by zooming.

(a) Random Order Scatterplot (b) Point Filtering (c) KDE (d) KDE and filtered points (e) Splatterplot

Fig. 12. Applying different scatter-data visualization techniques to a data set generated by sampling a differently located Gaussian for each of eight groups. Only splatterplots are able to convey the circular nature of each set, the degree of overlap, and the existence of outliers

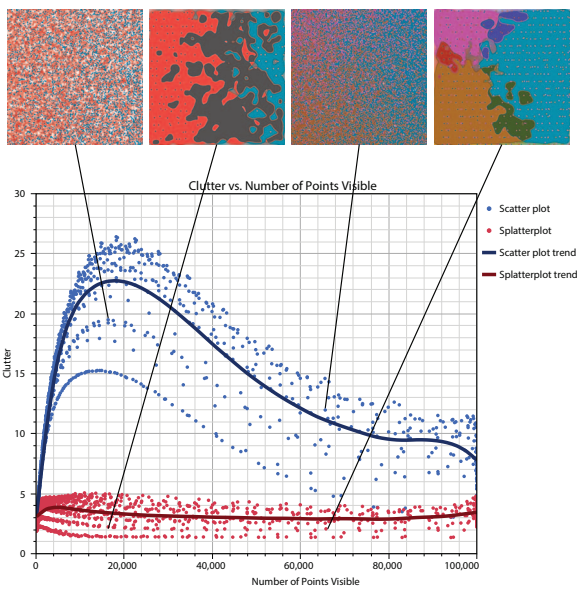


Fig. 10. Scatter plot of clutter measurements for scatter plots and splatterplots. Notice how the clutter of splatterplots is lower and varies little with number of points, while the clutter of scatter plots is higher and varies considerably with number of points shown.

outliers beyond the core dense region. We can also see that the method is beginning to break down visually with 8 groups: it is difficult to keep this many colors visually distinct.

5.2 Fatal Car Crashes

Here we show a dataset from the Fatality Analysis Reporting System (FARS). We picked two years, 2005 and 2010, and plotted the position of fatal car crashes. From figure 13 we can see that car crashes tend to cluster in more heavily populated areas. As we zoom in to the midwest region of the U.S. we can see different clusters rise up, corresponding to large cities such as Chicago and Indianapolis. The use of splatterplots enables us to see that these two data groups are very similar, but still highlights differences. Note that this example illustrates the use of scatterplots with a provided spatialization: the technique is appropriate to apply to maps.

5.3 Tree Cover Type Dataset

This standard dataset from the UCI machine learning repository [37] is derived from a cartographic survey of the Roosevelt National Forest of Colorado and contains over half a million instances, each with several different normalized cartographic measurements, separated into seven different kinds of tree cover types. A quick look at the summary statistics of the dataset reveals that two of the tree cover types, Lodgepole Pine and Spruce, are in much higher abundance than the others. One question we could ask is if there are any specific patterns and trends in the data that can help us decipher why this is. To explore this question we plot the top two cover types, against the bottom two, Cottonwood/Willow and Aspen.

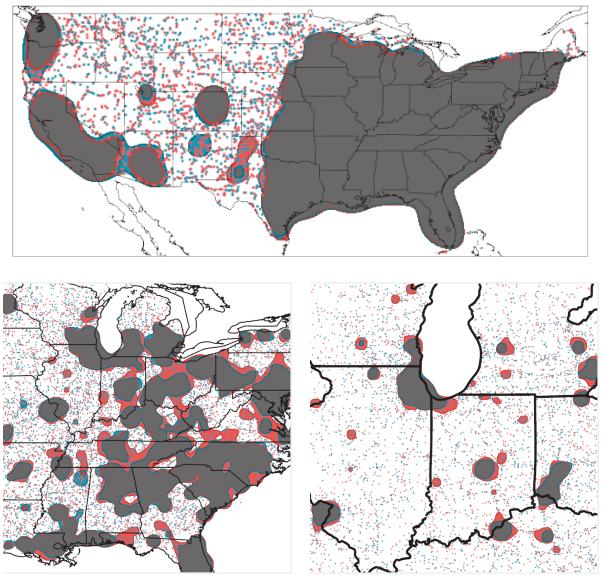
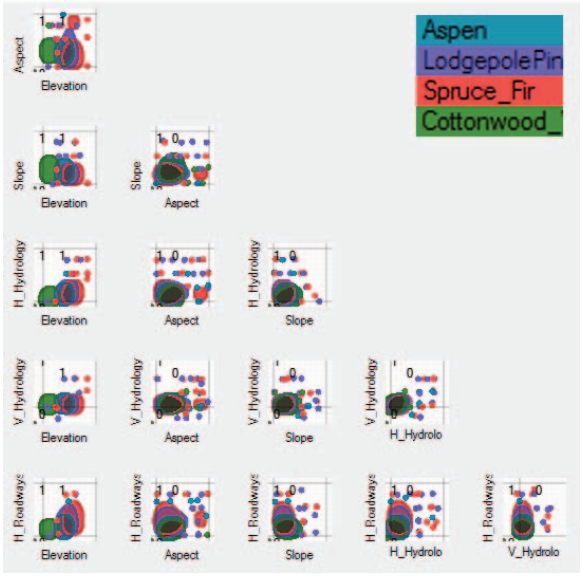


Fig. 13. A Splatterplot of fatal car crash locations in 2005 (blue) and 2010 (red). This demonstrates the technique working on a dataset with a spatial embedding. The views contrast dense and sparse regions allowing the similarities of the subgroups to be seen.

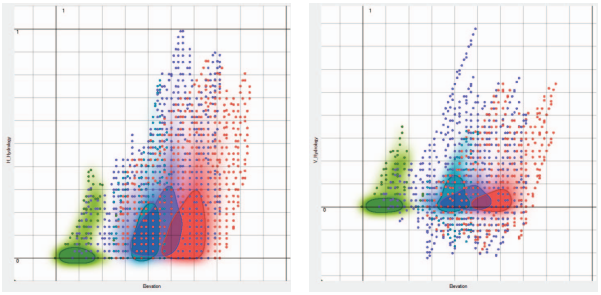


(a) SPLAM

Fig. 14. Splatterplot Matrix (SPLAM) of four different tree cover types. Notice how the elevation dimension tends to separate the data groups. Only the top left 6 variables are shown.

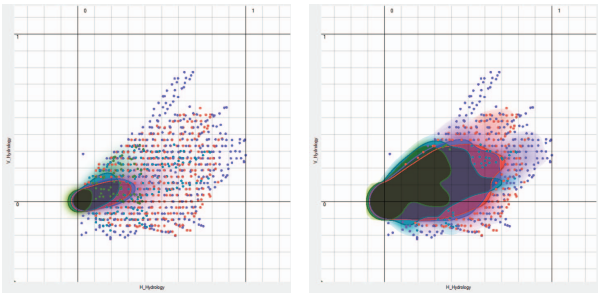
One of the advantages to using splatterplots is that even when the display size is small, we can still perceive general trends in the data. This is specially useful when looking at splatterplot matrices (Figure 14a).

An inspection of the Splatterplot matrix (SPLAM) reveals that elevation tends to spread out and separate



(a) H Hydrology (b) V Hydrology

Fig. 15. Large splatterplots of Elevation vs H Hyrdology and V Hydrology. The vertical stripes are an artifact of the data.



(a) Dense regions (b) Threshold most data

Fig. 16. Distance to water sources plotted against each other. From the splatterplots we can see that Cottonwood (green) likes to be close to water sources. Lodgepole and Spruce have a much higher tolerance for distance to water.

the four groups. Looking closely at the first column, we can also see that the ranges of the groups along H Hydrology (horizontal distance to nearest water source) varies. Figure 15a examines this more closely.

The large splatterplot confirms the judgement made from the small plot in the SPLAM. It also reveals the full extent of the data in each group. We can see that even though the dense regions of all the groups tend to have a pretty narrow range of H Hydrology, Lodgepole (purple) and Spruce (red) spread out much further than the other two cover types. If we also examine the plot of Elevation vs V Hydrology (Figure 15b), we can see that the same types of patterns emerge. In fact, if we plot H Hydrology vs V Hydrology another pattern becomes clear.

From Figure 16 we can see that Cottonwood and Apsen have a much smaller range of distances to water than Lodgepole and Spruce. From the analysis of the data we can start to form some hypothesis about why there are so many more Lodgepole and Spruce cover types than Cottonwood and Aspen. Cottonwood seems to like lower elevations, and be close to water sources. This could probably mean that Cottonwood require more moisture to grow, and are more fragile trees. Aspen trees seem to be a bit more tolerant

to water source distance, but seem particularly picky on elevation. This could be to the preference of a type of soil, or temperature. In contrast, while Lodgepole and Spruce seem to concentrate in specific regions elevation and water source distances, both of these tree types seem a lot more tolerant of large changes to these variables. One might hypothesize that Spruce and Lodgepole are sturdier, more adaptable trees. All of these findings suggest further questions deserving investigation.

5.4 Digital Humanities Datasets

This is a dataset consisting of over a thousand English texts from 1530 to 1800. Our domain collaborators have tagged each text using simple word matching that separate words into about 100 different categories. The counts of each of these types of words is then normalized by text length, creating a vector representation for each text. Each of these text has also been labeled with a genre. One of the things that our collaborators were interested in was how well texts from the same or similar genres clustered together. We performed principal component analysis (PCA) on the data, and plotted the first two components against each other [38].

As stated before, our splatterplot visualization allows us to make judgements about data even when dealing with small plots, such as in a SPLAM. Another type of display that is useful when there are many different groups at once is a one versus the rest type of display (Figure 17).

Figure 17 shows the top nine genres, and lumps the rest of the texts into a group labeled "Rest". Each of these groups is then plotted along with the rest of the data corpus. From the figure, we can see that save for the "Rest" group, most of the genres tend to group fairly well. We can also plot the top seven genres in the same plot and see how they relate to each other (Figure 18).

The first principal component (x-axis) separates the genres into two larger groups. On the left we have Sermons, Religious Prose, and Nonfictional Prose, while on the right we have Fictional Prose, Verse Collection, Poetry, and Drama. At a first glance, it would seem that Prin1 tends to separate works of fiction from nonfiction. Prin2 also has interesting properties, as it tends to separate genres on the right, but not those on the left. Looking at the loadings of the original data dimensions along these two principal components would open a line of inquiry and analysis to further investigate this dataset.

6 Conclusion

Splatterplots provide a new technique for displaying point data that scales well with the number of points. As the number of points grows larger, the amount of information to be presented exceeds what can be displayed. Splatterplots explicitly abstract the data to fit within a screen-space information density bound. By showing dense regions of points as contour-bounded filled areas, and sub-sampling the number of points outside these areas, Splatterplots preserve the ability to see overall shapes and trends, relationships between sets, and a sense of the range of outliers – even as the number of data points greatly exceeds the number of pixels. Detailed information hidden by these abstractions can be revealed through interactive navigation, made possible at interactive rates through an efficient GPU-accelerated implementation.

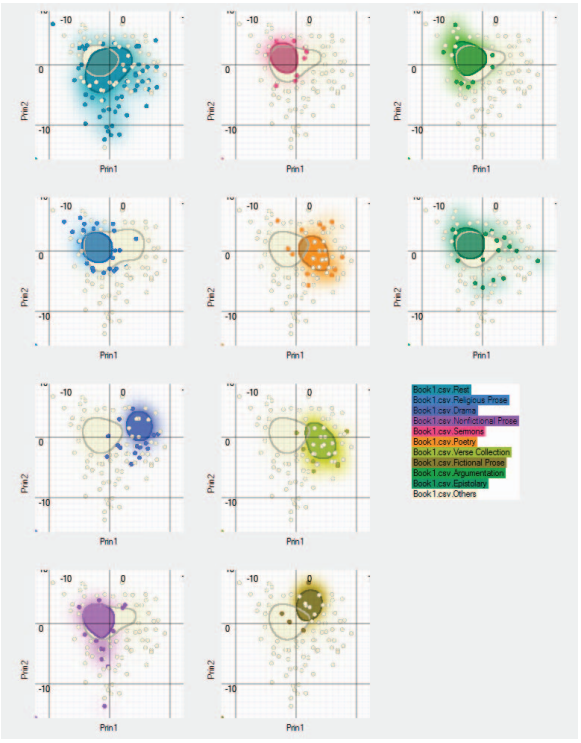
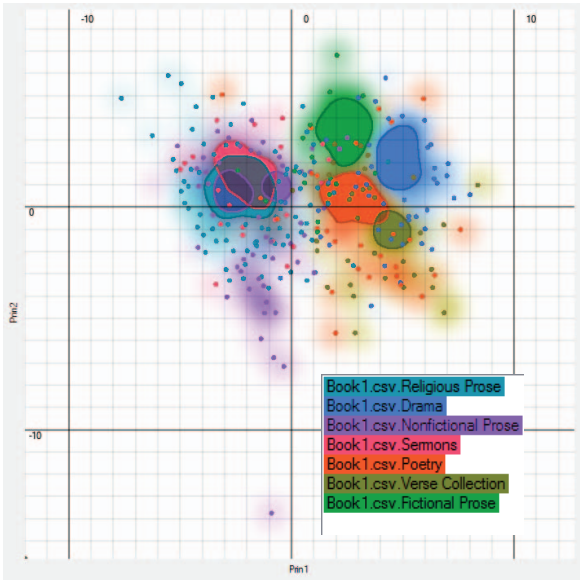
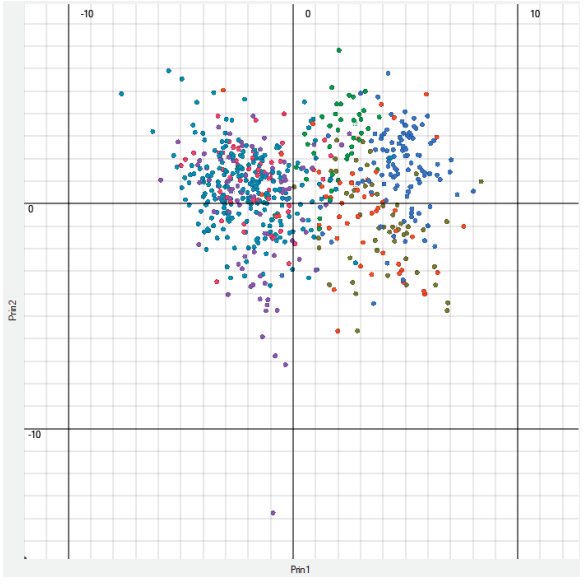


Fig. 17. Series of splatterplots that show many groups, each at a time, against the rest of the ungrouped data.

The key feature of Splatterplots, that information is abstracted based on screen-space limits to enforce readability, is also a limitation. Abstraction removes detail that may be important to the viewer. The choices in Splatterplots emphasize conveying shape and set relations, at the expense of providing details of density and specific point positions. Specific points can be revealed through zooming. However, for some applications, different tradeoffs may be advantageous. Splatterplots allow the user to explore the density through interactive control of the aggregation parameter. However, user control over parameters emphasizes another limitation: changes to the parameter effect the shapes of the dense regions. The impact of this issue is lessened because Splatterplots have few parameters; these parameters are easy to control because they are defined in screen (rather than data)



(a) Splatterplot

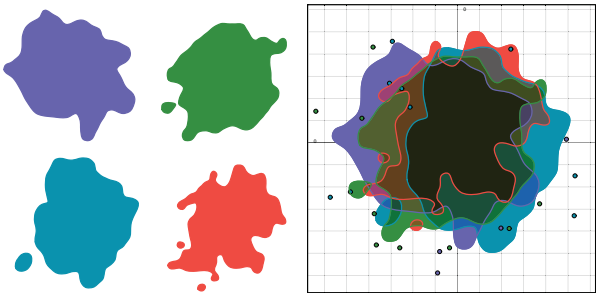


(b) Scatter plot

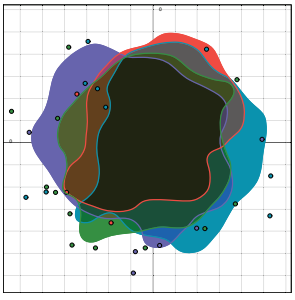
Fig. 18. Top seven genres plotted at once. Notice how Prin1 separates the genres into two larger groups with Sermons,Religious Prose, and Nonfictional Prose on the left and the other 4 genres on the right. These trends are more difficult to pick out in the scatter plot.

space; and the displays can be updated at interactive rates allowing experimentation with parameters.

Another limitation of Splatterplots is that most abstraction is done on individual data subgroups, which may potentially lead to visual complexity due to the interactions between subgroups. Figure 19 shows an example of this phenomenon. Notice how in Figure 19a each of the separate contours is relatively smooth and has easy to perceive features. However, once the contours are displayed on the same plot, the contours interact in non obvious ways, creating a large number



(a) individual contours (b) combined contours



(c) increased smoothing

Fig. 19. Adding more sets to a single plot has the effect of increasing visual complexity in the final view.

of visually salient and distracting features. Increasing the amount of smoothing during KDE greatly diminishes and performing shape optimization reduce these problems and makes the resulting Splatterplot more comprehensible. However, we rely on user tuning of the abstraction parameters to find the appropriate balance of data fidelity and readability.

While Splatterplots scale well in the number of points, they are limited in their scaling in the number of subgroups. Our reliance on colors as the primary group identification mechanism limits the number of potential groups, especially when a combinatorial explosion of new groups are formed from the group intersections. In situations where many overlapping groups must be displayed, alternate identification mechanisms, such as textual labels, will be needed. Our experience suggests that the visual complexity of multiple overlapping sets is the main limiting factor.

Our prototype implementation of Splatterplots has proven effective at displaying point sets with moderate numbers of sub-groups (5-8), and large numbers of points (several million). By aggregating dense regions and subsampling sparse ones, Splatterplots quickly convey the overall distributions of large data sets.

ACKNOWLEDGMENTS

This project was supported in part by NLM training grant 5T15LM007359 and NSF awards CMMI-0941013, and IIS-1162037. Domain collaborators were supported in part by a Mellon Foundation Grant. We thank Robin Valenza and the members of the UW Graphics Group for their help in preparing this paper.

REFERENCES

[1] M. Harrower and C. a. Brewer, "ColorBrewer.org: An Online Tool for Selecting Colour Schemes for Maps," *The Cartographic Journal*, vol. 40, no. 1, pp. 27–37, Jun. 2003.

[2] M. Theus, "Scaling Up Graphics," in *Graphics of Large Datasets: Visualizing a Million*, A. Unwin, M. Theus, and H. Hofmann, Eds. New York: Springer, 2006, ch. 3, pp. 56–72.

[3] B. Trumbo, "A theory for coloring bivariate statistical maps," *The American Statistician*, vol. 35, no. 4, pp. 220–226, Nov. 1981.

[4] P. Rheingans, "Task-based color scale design," in *PROC SPIE INT SOC OPT ENG*, vol. 3905, 2000, pp. 35–43.

[5] C. Ware, "Quantitative texton sequences for legible bivariate maps," *IEEE Trans. Computer Graphics and Visualization*, vol. 15, no. 6, pp. 1523 –1530, 2009.

[6] J. R. Miller, "Attribute blocks: Visualizing multiple continuously defined attributes," *IEEE Computer Graphics and Applications*, vol. 27, pp. 57–69, 2007.

[7] H. Hagh-Shenas, S. Kim, V. Interrante, and C. Healey, "Weaving versus blending: a quantitative assessment of the information carrying capacities of two alternative methods for conveying multivariate data with color." *IEEE Trans. Computer Graphics and Visualization*, vol. 13, no. 6, pp. 1270–7.

[8] C. Collins, G. Penn, and S. Carpendale, "Bubble sets: revealing set relations with isocontours over existing visualizations." *IEEE Trans. Computer Graphics and Visualization*, vol. 15, no. 6, pp. 1009–16, 2009.

[9] P. Simonetto, D. Auber, and D. Archambault, "Fully automatic visualisation of overlapping sets," *Computer Graphics Forum*, vol. 28, no. 3, pp. 967–974, 2009.

[10] G. Ellis and A. Dix, "Density control through random sampling: an architectural perspective," in *IEEE Information Visualisation*. IEEE, 2002, pp. 82–90.

[11] A. Dix and G. Ellis, "by chance enhancing interaction with large data sets through statistical sampling," in *Proceedings of the Working Conference on Advanced Visual Interfaces*. ACM, 2002, pp. 167–176.

[12] E. Bertini and G. Santucci, "Give chance a chance: modeling density to enhance scatter plot quality through random data sampling," *Information Visualization*, vol. 5, no. 2, 2006.

[13] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," *IEEE Trans. Computer Graphics and Visualization*, vol. 13, no. 6, pp. 1216–1223, 2007.

[14] T. N. Dang, L. Wilkinson, and A. Anand, "Stacking graphic elements to avoid over-plotting," *IEEE Trans. Computer Graphics and Visualization*, vol. 16, no. 6, pp. 1044 –1052, 2010.

[15] M. Tory, D. Sprague, F. Wu, W. Y. So, and T. Munzner, "Spatialization design: comparing points and landscapes." *IEEE Trans. Computer Graphics and Visualization*, vol. 13, no. 6, pp. 1262–9, Jan. 2007.

[16] D. Keim, M. Hao, U. Dayal, H. Janetzko, and P. Bak, "Generalized scatter plots," *Information Visualization*, vol. 9, no. 4, pp. 301–311, 2010.

[17] M. Hao, U. Dayal, R. Sharma, D. Keim, and H. Janetzko, "Variable binned scatter plots," *Information Visualization*, vol. 9, no. 3, pp. 194–203, 2010.

[18] N. Elmqvist, P. Dragicevic, and J.-D. Fekete, "Rolling the dice: Multidimensional visual exploration using scatterplot matrix navigation," *IEEE Trans. Computer Graphics and Visualization*, vol. 14, no. 6, pp. 1141–1148, 2008.

[19] M. Sips, B. Neubert, J. Lewis, and P. Hanrahan, "Selecting good views of high-dimensional data using class consistency," in *Computer Graphics Forum*, vol. 28, no. 3. Wiley Online Library, 2009, pp. 831–838.

[20] L. Wilkinson and G. Wills, "Scagnostics distributions," *Journal of Computational and Graphical Statistics*, vol. 17, no. 2, pp. 473–491, 2008.

[21] R. Rensink and G. Baldridge, "The perception of correlation in scatterplots," in *Computer Graphics Forum*, vol. 29, no. 3. Wiley Online Library, 2010, pp. 1203–1210.

[22] J. Li, J. Martens, and J. van Wijk, "A model of symbol size discrimination in scatterplots," in *Proceedings of the 28th international conference on Human factors in computing systems*. ACM, 2010, pp. 2553–2562.

[23] J. Li, J. van Wijk, and J. Martens, "A model of symbol lightness discrimination in sparse scatterplots," in *Pacific Visualization Symposium (PacificVis), 2010 IEEE*. IEEE, 2010, pp. 105–112.

[24] ——, "Evaluation of symbol contrast in scatterplots," in *Pacific Visualization Symposium (PacificVis), 2009 IEEE*. IEEE, 2009, pp. 97–104.

[25] S. Bachthaler and D. Weiskopf, "Continuous scatterplots," *IEEE Trans. Computer Graphics and Visualization*, vol. 14, no. 6, pp. 1428–1435, 2008.

[26] J. Wolfe, K. Kluender, D. Levi, L. Bartoshuk, and R. Herz, *Sensation & Perception*. Sinauer Associates Inc, 2011.

[27] S. Dakin and N. Baruch, "Context influences contour integration," *Journal of Vision*, vol. 9, no. 2, 2009.

[28] J. Elder and R. Goldberg, "Ecological statistics of gestalt laws for the perceptual organization of contours," *Journal of Vision*, vol. 2, no. 4, 2002.

[29] S. Kuai and C. Yu, "Constant contour integration in peripheral vision for stimuli with good gestalt properties," *Journal of Vision*, vol. 6, no. 12, 2006.

[30] M. Fairchild, *Color appearance models*, ser. Wiley-IS&T series in imaging science and technology. J. Wiley, 2005.

[31] J. Thompson and R. Tapia, *Nonparametric function estimation, modeling, and simulation*, ser. Miscellaneous Bks. Society for Industrial and Applied Mathematics, 1990.

[32] D. W. Scott, *Kernel Density Estimators*. John Wiley & Sons, Inc., 2008, pp. 125–193.

[33] G. Rong and T. Tan, "Jump flooding in gpu with applications to voronoi diagram and distance transform," in *Proceedings of the 2006 symposium on Interactive 3D graphics and games*. ACM, 2006, pp. 109–116.

[34] A. Sarkar, L. Blondé, P. Le Callet, F. Autrusseau, P. Morvan, J. Stauder *et al.*, "A color matching experiment using two displays: design considerations and pilot test results," in *Proceedings of the Fifth European Conference on Color in Graphics, Imaging and Vision*, 2010.

[35] R. Rosenholtz, Y. Li, and L. Nakano, "Measuring visual clutter," *Journal of Vision*, vol. 7, no. 2, 2007.

[36] R. Rosenholtz, A. Dorai, and R. Freeman, "Do predictions of visual perception aid design?" *ACM Transactions on Applied Perception*, vol. 8, no. 2, pp. 1–20, Jan. 2011.

[37] A. Frank and A. Asuncion, "UCI machine learning repository," 2010. [Online]. Available: http://archive.ics.uci.edu/ml

[38] M. Correll, M. Witmore, and M. Gleicher, "Exploring collections of tagged text for literary scholarship," *Computer Graphics Forum*, vol. 30, no. 3, pp. 731–740, jun 2011.

Adrian Mayorga is a graduate student in the Department of Computer Sciences at the University of Wisconsin-Madison working under Professor Michael Gleicher. His research has focued in the use of abstaction to enhance visualizations of large amounts of data. Prior to joining the University of Wisconsin, Adrian recevied his B.S. in Computer Science from the University of Texas at Austin.

Michael Gleicher is a Professor in the Department of Computer Sciences at the University of Wisconsin, Madison. He is founder of the Department's Computer Graphics group. His research interests span the range of visual computing, including data visualization, image and video processing tools, and character animation techniques for films and games. Prior to joining the university, Prof. Gleicher was a researcher at The Autodesk Vision Technology Center and in Apple Computer's Advanced Technology Group. He earned his Ph. D. in Computer Science from Carnegie Mellon University and a B.S.E. in Electrical Engineering from Duke University. Prof. Gleicher is an ACM Distinguished Scientist.