Data Sampling in Multi-view and Multi-class Scatterplots via Set Cover Optimization

Ruizhen Hu, Tingkai Sha, Oliver van Kaick, Oliver Deussen, and Hui Huang

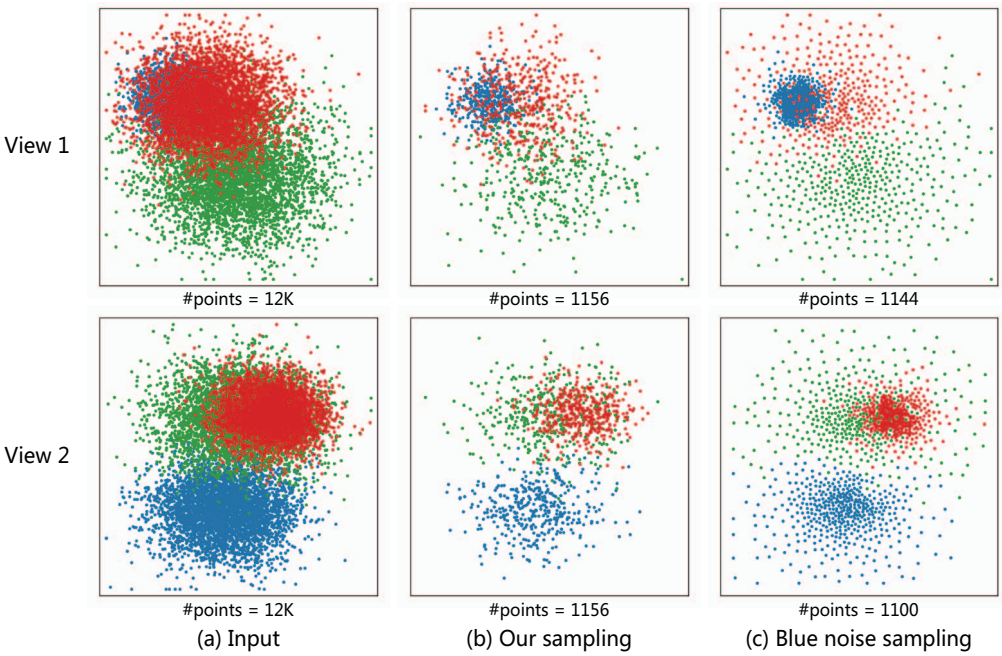


Fig. 1. Our method creates a sub-sampled point set that is optimized for different views of a SPLOM and per class. In contrast to blue noise scatterplot sampling (and other sampling methods), which create different point sets for different views, our method uses joint optimization to yield a single point set for multiple views. This way, it not only optimizes for multi-view and multi-class scatterplots simultaneously, but also presents results perceptually similar to the original data distributions while reducing overdraw.

Abstract—We present a method for data sampling in scatterplots by jointly optimizing point selection for different views or classes. Our method uses space-filling curves (Z-order curves) that partition a point set into subsets that, when covered each by one sample, provide a sampling or coreset with good approximation guarantees in relation to the original point set. For scatterplot matrices with multiple views, different views provide different space-filling curves, leading to different partitions of the given point set. For multi-class scatterplots, the focus on either per-class distribution or global distribution provides two different partitions of the given point set that need to be considered in the selection of the coreset. For both cases, we convert the coreset selection problem into an Exact Cover Problem (ECP), and demonstrate with quantitative and qualitative evaluations that an approximate solution that solves the ECP efficiently is able to provide high-quality samplings.

Index Terms—Sampling, Scatterplot, SPLOM, Exact Cover Problem

1 Introduction

Scatterplots are an important method for displaying high-dimensional data. A scatterplot shows such data from a single view direction, capturing two selected dimensions of the data points. In a scatterplot matrix (SPLOM), views from all main directions are combined. If the dimensionality is high, such matrices can easily contain hundreds of plots, therefore methods for selecting interesting views have been developed [34]. Scatterplots can also be used to display data that has already been reduced in its dimensionality [23], where the most prominent aspects of such data are revealed when visualized [32].

A large problem for scatterplots is overdraw, when too many points are drawn onto a region of the plot. Many methods have been proposed to alleviate this, such as methods that alter the transparency, position, density, size, color or form of the markers that are associated with each point, even animating them. Overviews about such possibilities and associated quality metrics are given by Bertini et al. [7] and Behrisch et al. [3]. The problem becomes even more severe when points are labeled and belong to different classes. Such multi-class scatterplots need to be visualized with additional visual cues such as different colors or shapes for the representation of points of different classes.

- *R. Hu, T. Sha, and H. Huang are with Shenzhen University, Visual Computing Research Center, China. E-mail:{ruizhen.hu, shatingkai, hhzhiyan}@gmail.com.*
- *O. van Kaick is with Carleton University, School of Computer Science, Canada. E-mail: ovankaic@gmail.com.*
- *O. Deussen is with Konstanz University, Germany and Shenzhen VisuCA Key Lab, SIAT, China. E-mail: oliver.deussen@uni-konstanz.de.*
- *Hui Huang is the corresponding author of this paper.*

Manuscript received 31 Mar. 2019; accepted 1 Aug. 2019.
Date of publication 16 Aug. 2019; date of current version 20 Oct. 2019.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TVCG.2019.2934799

Overdraw of the scatterplots can also be alleviated by subsampling the input set of points to reduce clutter in the plot. The classical way of doing this is random sampling, where points are selected by a random selection process. An improvement of this basic method is non-uniform sampling [4] that treats different parts of the scatterplot in different ways. This process can be further refined by considering human perception, by ensuring that important aspects of the plots such as cluster densities are seen by the viewer [6]. Chen et al. [9] employ a multi-class blue noise sampling method [39], where points are selected from the original plot in a way that an approximation of a blue-noise point distribution is achieved. This way a smooth representation is created, but local patterns are potentially missed.

In this paper we describe a new method for sampling scatterplots. Our main contribution is to perform the sampling simultaneously for different view directions. Our method uses a local binning strategy proposed by Zheng et al. [42, 43] to select a proper fraction of the points. In their work, a space filling curve (Z-order curve) is employed to create bins and one point from each bin is then selected to represent the entire point set. Such a set of sample points is called a *coreset*, since it provides approximation guarantees in relation to the original point set. For more than a single projection, a space filling curve and coreset are produced for each view. In our work, we perform the selection of coresets simultaneously for all views. Specifically, our goal it to sample a coreset that considers the Z-order curves of all the individual views. We show that the selection of the coreset converts into an *Exact Cover Problem* (ECP) that is NP-complete [18], but can be solved efficiently using heuristics [37].

In a similar way, multi-class sampling can be approached. When multiple classes exist, we need a sampling strategy that is optimal for each class and also for the distribution of all points. Therefore, we obtain the Z-order for each class and the entire set, and then compute an optimal set cover by considering all the point orders.

We show with an experimental evaluation that our method has several advantages in relation to previous works. By computing a sampling simultaneously for multiple views, we only need to store a single sampling for any large scatterplot matrix, rather than an individual sampling for each view. Moreover, the samplings can be computed efficiently, e.g., a few seconds for a set of 20K points. Finally, in comparison to previous works, the samplings obtained with our method better reflect the density of points in the original dataset, while also reducing overdraw problems and providing better class separation in the multi-class setting (Figure 1).

In summary, the main contribution of this paper includes:

- A joint optimization framework for subsampling multiple views in SPLOMs and multi-class scatterplots;
- A formulation of SPLOM and multi-class scatterplot sampling in terms of the ECP, which can be extended for outlier inclusion;
- An efficient greedy algorithm that solves the ECP approximately, with experimental evaluations and user studies to demonstrate that the scatterplots obtained with our sampling are more informative than those generated with existing methods.

2 Related Work

2.1 Overdraw reduction for scatterplots

As mentioned above, quite a few methods have been proposed to alter the graphical representation of the points in a scatterplot to reduce overdraw.

A straightforward approach is to reduce the size of the markers that represent the points [24, 41], to change the shapes of markers [22], or to make them translucent in order to allow the user to see through [26, 40]. Recent publications alter different visual variables at the same time [28] and optimize opacity, size, as well as aspect ratio of the plot based on visual quality metrics.

For multi-class scatterplots, in addition, the class separability has to be taken into account. Luboschik et al. [25] use weaving to show overlapping regions of different classes. Wang et al. [38] find the best color assignment from a given palette to optimize class separability.

Instead of altering the markers, other approaches try to relocate them by shifting points along space-filling curves to unoccupied positions [20]. So called "generalized scatterplots" [19] use a non-linear warping scheme to enable users to control overdraw.

Yet another approach is to convert the scatterplots into continuous density fields [8, 36] and then show these fields by color-coded density plots or by contour lines. Novotny and Helwig [29] use multi-dimensional binning to compute local densities and use this for creating the fields, while Bachthaler and Weiskopf [2] use specialized interpolation methods. Feng et al. [16] use kernel density estimation [33] to generate density plots. Mayorga and Gleicher [27] combine contour drawings with outliers as discrete elements.

All these methods work with all the points of a scatterplot while sampling methods try to find a subset of the points for representing the original dataset [15]. This is a non-trivial process since outliers and rare point classes have to be represented, while the density of the other points has to be reduced massively while preserving their relative densities. Random sampling [13, 14] is a simple solution for that, but fails to treat different densities differently, which can be improved by using non-uniform sampling [4]. Bertini and Santucci [5] present an automatic method to preserve relative densities and to reduce visual clutter, which had already been introduced by Tufte [35]. Chen et al. [9] construct a density plot using kernel density estimation and then apply multi-class blue noise sampling to show the points in a reduced form.

Our method is based on the Z-order curve method [42, 43] to find a coreset for a given set of points. Here, a space filling curve is used to define bins to which points are assigned. For sub-sampling, one point from each bin is used to represent the entire plot. We extend this method for multiple views and classes by formulating the problem as an Exact Cover Problem (ECP) [18], which can be solved approximately in an efficient way using a greedy strategy [37].

2.2 Overdraw alleviation for SPLOMs

Although there have been many previous works focusing on reducing overdraw in a single scatterplot either with single-class or multi-class data, very few works have been proposed to handle the overdraw problem for scatterplot matrices. Furthermore, it is unclear how to extend methods that have been proposed for individual scatterplots to SPLOMs with multi-view consistency. Bertini and Santucci [6] develop uniform and non-uniform sampling strategies for single scatterplots, but their framework for measuring the degradation cannot easily be adapted to SPLOMs. While sampling methods such as Chen et al. [9] create interesting results for multi-class scatterplots, it stays unclear how to extend them to SPLOMs, since one could use a single point set for all views or also optimize different point sets for the different views.

A recent work [10] explores the use of animation to address the problem of overdraw and identify regions of varying density and diversity in multi-class SPLOMs. However, animation tends to be distracting in certain cases and has only a limited scope.

3 Data Sampling Methods

In this section, we describe our scatterplot data sampling methods. We first introduce our approach for simultaneously sampling data points across multiple scatterplots. We then describe how to apply this method to multi-class scatterplots and how to enable outlier inclusion as well as use view selection for optimizing the sampling process.

3.1 Simultaneous sampling of points for SPLOMs

Given a scatterplot matrix (SPLOM), our goal is to select a set of sample points that simultaneously reduce overdraw for all the views of the SPLOM, improving the visualization of each individual plot. First, for each view, we use the Z-order method to extract subsets of points that, when covered by a sampling, provide a good approximation of the view in terms of a kernel density estimate (KDE) of the points [42]. Next, we obtain a sampling that simultaneously provides a good approximation for all the views by solving an exact cover problem, that is, our goal is to select a set of points that covers all the subsets. In the following, we first describe the method for computing the Z-order of a point set, and

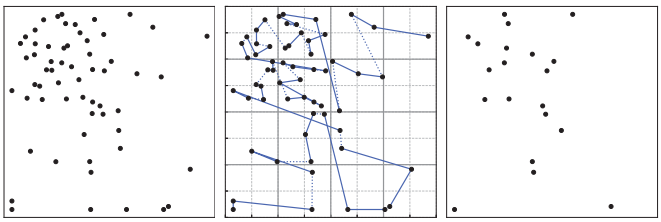


Fig. 2. Sampling based on Z-order: (a) input set of points; (b) quadtree subdivision and z-order curve of the point set, where points inside the same subset are connected with solid lines (here each subset consists of 3 points); (c) final sampling result, composed of only one point from each subset.

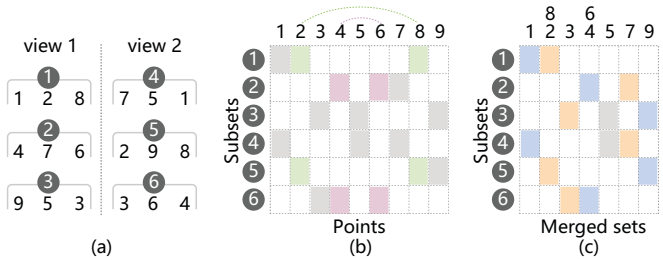


Fig. 3. Reduction of simultaneous point sampling for multiple views into a set cover problem. For each view in (a), the Z-order algorithm partitions the input point set into n subsets, where $n = 3$ in this example. From the partition, we can define a set cover matrix (b), where an entry (i, j) is non-zero (not white) if subset i contains point j. Our goal then is to select a set of points (columns) so that all the subsets (rows) are covered without redundancy. Given that certain points cover the same subsets (columns with the same color in (b)), we can merge similar points (c) and simplify the set cover problem. In this example, we can find two different covers indicated by the two different colors (blue and yellow).

then explain how we simultaneously optimize the sampling using such Z-order sets for multiple views.

3.2 Z-order sampling of a point set

The Z-order method is based on the creation of a space filling curve that defines a 1D order for a set of 2D points while at the same time preserving the spatial locality of the points reasonably well [43]. Suppose that we are given a point set P as input, with $|P| = N$, and our goal is to obtain a subsampled set $Q \subset P$ with $|Q| = n$. First, the 2D space covered by P is subdivided into four subdomains according to a quadtree structure. Next, a curve in the form of the letter "Z" orders the domains of the quadtree in a linear order. The procedure is recursively applied so that the quadtree adapts to the specific set of points and the curve defines an order for all the points. Then, given the points in the Z-order, we are able to sub-sample a set of points Q by selecting a specified number of points from each level of the quadtree (see Figure 2 for an illustration).

More precisely, suppose w.l.o.g. that all the 2D points in the input set P are normalized to the range $[0,1]^2$. Then, a quadtree subdivides this space into four subdomains or cells: $c_1 = \left[0, \frac{1}{2}\right] \times \left[0, \frac{1}{2}\right]$, $c_2 = \left[\frac{1}{2}, 1\right] \times \left[0, \frac{1}{2}\right]$, $c_3 = \left[0, \frac{1}{2}\right] \times \left[\frac{1}{2}, 1\right]$, $c_4 = \left[\frac{1}{2}, 1\right] \times \left[\frac{1}{2}, 1\right]$. The Z-order curve visits these cells in the order c_1, c_2, c_3, and c_4. When visiting a cell, the procedure is recursively applied: the cell is divided into four smaller cells which are also visited in Z-order. By performing the recursion until a cell contains a single point, we can extract a linear ordering for all the points in the input set P.

Moreover, to generate the sampling Q with n points, we randomly select a point $q_i \in Q$ from the rank range $\left[(i-1)\frac{|P|}{n}, i\frac{|P|}{n}\right]$, as illustrated in Figure 2(b)-(c). The sampling takes $O(|P| \log |P|)$ time. The motivation for using this specific sampling procedure is that it guarantees that Q approximates well the properties of P. To describe the theoretical guarantees of the sampling, we briefly discuss the use of KDEs for approximating a continuous density as follows.

3.3 Kernel density estimation

A kernel density estimation (KDE) is a non-parametric statistical representation of a continuous density, based only on a discrete sample of points and a kernel function. Given a set of points $P \subset \mathbb{R}^2$, the density estimate at any query point $x \in \mathbb{R}^2$ is given by:

$$\mathrm{KDE}_P(x) = \frac{1}{|P|} \sum_{p \in P} K_h(p,x), \tag{1}$$

where K_h is a kernel function $K_h : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ with bandwidth h. In our experiments, K_h is the Gaussian kernel with standard deviation σ:

$$K_h(p,x) = \frac{1}{\sigma h \sqrt{2\pi}} \exp\left(\frac{-\|p-x\|^2}{2\sigma^2 h^2}\right), \tag{2}$$

where $\sigma = 1$ and h is determined by Silverman's rule of thumb [33].

The KDE can be thought of as a smooth histogram which is centered at each query point and does not depend on bin centers and sizes, thus making it a more robust tool for the analysis of a distribution. Previous work has shown that we can approximate a KDE_P by using a *coreset* Q, which is a small but carefully selected sample of P that approximates P's properties as accurately as possible [42, 43]. Specifically, a coreset is a subset $Q \subset P$, such that $|Q| \ll |P|$ and for some error threshold ε:

$$\|\mathrm{KDE}_P - \mathrm{KDE}_Q\|_\infty = \max_{x \in \mathbb{R}^2} |\mathrm{KDE}_P(x) - \mathrm{KDE}_Q(x)| \leq \varepsilon, \tag{3}$$

implying that the maximum error of the approximation provided by the coreset is always bound by ε for any point x. The Z-order is one method to create such a coreset, and thus we use it in our work due to these theoretical guarantees. Specifically, the Z-order sampling builds a coreset of size $n = O(\frac{1}{\varepsilon} \log^{2.5} \frac{1}{\varepsilon})$ that creates an *ε-approximation*.

3.4 Sampling coresets for SPLOMs

The Z-order combined with random sampling provides an optimal subset of points for drawing one view of a SPLOM. However, as the distribution of points changes for different views, we need to repeat this construction and store a different coreset for every scatterplot of a SPLOM, which can be prohibitive for high-dimensional datasets and cannot guarantee the consistency of point selection between different views. Thus, our main contribution is to introduce a method for computing a coreset that simultaneously serves all the views of a SPLOM. Moreover, as we will discuss in Section 3.7, the coreset can also be computed only for selected views of a SPLOM.

Specifically, our goal is to build a coreset Q (with $|Q| = n$) for a high-dimensional input set of points $P \in \mathbb{R}^{N \times d}$ and a SPLOM with m views. Typically, for a complete SPLOM, $m = d \times d$, although we only need to consider the $m = d \times (d-1)/2$ views determined by the unique combinations of different dimensions. Each view v projects the set P onto 2D based on two selected dimensions of the points, which we denote $\pi_v(P) = P' \in \mathbb{R}^{N \times 2}$. Suppose that we computed a coreset for a specific view P', one of our key observations is that the Z-order algorithm splits P' into n disjoint subsets (the rank ranges defined by $\left[(i-1)\frac{|P|}{n}, i\frac{|P|}{n}\right]$, where $i \in \{1, \ldots, n\}$). The algorithm then randomly samples one point from each subset to provide a subsampling with n points. Thus, any other point in a subset S would be a good representative sample of S, as the main requirement for guaranteeing the error bounds of the coreset is that the coreset should contain one point from each subset. We use this flexibility in the selection of points to turn the problem of selecting a coreset for all the views into an exact cover problem.

For each view, the Z-order algorithm provides a partition of P into n subsets. However, as illustrated in Figure 3(a), the partition can be different for each view. In particular, by computing the Z-order for the entire SPLOM, we obtain $m \times n$ subsets. Our goal is then to select

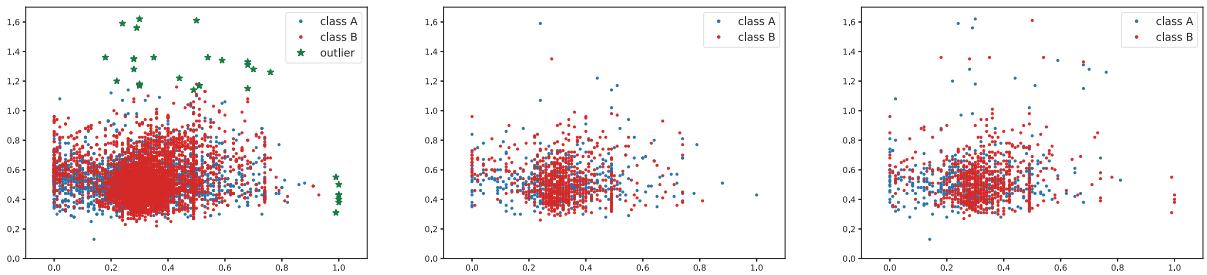


Fig. 4. Sampling with outlier inclusion. Original scatterplot (a) with two classes (blue and red), where outliers are marked with green stars. The samplings (b) and (c) both cover each subset of the individual views (per-class and global view), but (c) selects the outliers of the subsets.

a coreset Q which contains at least one point from each of the $m \times n$ subsets. In this manner, we will cover all the subsets of all the partitions and ensure that the error bound of Eq. (3) is met simultaneously for all the views. Thus, we can turn the selection of the coreset Q into an exact set cover problem: our goal is to select a set of points Q such that each subset is covered by one point, as illustrated in Figure 3(b). When representing the set cover problem as a matrix, our goal is to select a subset of columns (points) which together have exactly one 1 in each row (we cover each subset without redundancy).

Moreover, given that some points cover the same subsets, we can simplify the set cover matrix by merging the coincident points, as illustrated in Figure 3(b)-(c). If a merged set of points is selected for the final cover, the set cover includes multiple solutions and gives us some flexibility in selecting the final points that compose the coreset.

Given that the exact set cover problem is NP-hard, we compute the solution with the greedy heuristic method of Chvatal [11], which has the best approximation ratio known: the cost of the greedy solution is at most $H(L)$ times the cost of an optimal cover, where $H(k) = \sum_{j=1}^{k} \frac{1}{j}$ and L is the size of the largest set in the problem. A trivial implementation of the heuristic has a worst-case $O(mnN^2)$ time complexity, for N points and mn subsets, although it runs much faster in practice. Moreover, we may not always be able to find an exact set cover. In this case, the heuristic is also useful in providing an approximate set cover, i.e., some subsets may be covered by more than one point. Thus, a second goal of the heuristic is to minimize the redundancy in the approximate cover.

Regarding the theoretical guarantees of the algorithm, by covering each subset with exactly one point, the Z-order guarantees that the KDE error (Eq. 3) for each view is bounded by ε. Since our heuristic can select more than one point per subset, we cannot guarantee that the bound is preserved anymore. However, as we show in Section 4, in practice, small KDE errors are obtained with our method, since the heuristic attempts to minimize the redundancy in the set cover. Moreover, the number of points sampled by our method may be larger than a specified n. However, by assigning a cost of 1 to each point to be selected, with the analysis of the heuristic we see that the number of points is at most $H(L)$ times the number of points in an optimal cover.

Note that, recent theoretical work has described algorithms in high dimensions for sampling near-optimal coresets of KDEs, which provide guaranteed bounds and may provide better coreset sizes [30]. However, these methods are unpractical for large datasets as they depend on an analysis of a Gram matrix of the kernel applied to the entire point set, which can require significant time and space resources, especially if the matrix needs to be decomposed.

The idea of computing a set cover to integrate multiple samplings is quite versatile, and thus we also use this method as a solution for other SPLOM-related tasks, described as follows.

3.5 Sampling multi-class scatterplots

Another problem related to SPLOM visualization is to obtain a scatterplot for a multi-class point set, where each point is associated to a class label $l \in \mathcal{N}$, with $\mathcal{N}$ being the full set of labels. In this case, our goal is to provide a sampling that is optimal for each label as well as for the entire point set (the global view of the points). We thus obtain the Z-order for the points of each label and also for the entire point set, and then compute a set cover to extract a single sampling that best approximates the properties of all classes and the global view. Note that, since the classes are disjoint, the ordering of all the classes results in a collection of subsets that provides "one view" of the data. Thus, computing the sampling in the multi-class case is equivalent to applying our method to two views (one for the collection of the class subsets and another for the entire set).

3.6 Outlier inclusion

Although selecting a single point from each subset provides a subsample with approximation guarantees, the user may also want to explore outliers in the data. Thus, we use the flexibility provided by the merged sets in the selection of the set cover that provides us with the final point sample. First, we use a distance-based outlier detection method to tag certain points as outliers [21]. More specifically, given a search radius r and a threshold θ, with $(0 < \theta < 1)$, a point p is considered to be an outlier if less than $\theta \times N$ points are found inside a neighborhood with radius r centered at the point p. Next, points tagged as outliers are given priority to be selected when a merged set is included in the computed cover. A comparison of the default sampling with a version that handles outliers is given in Figure 4.

3.7 View selection for sampling optimization

As discussed above, a complete SPLOM typically contains $m = d \times (d-1)/2$ different views that we are interested in. When d is large, the resulting matrix can contain a large number of views, and thus optimizing the sampling for all the views can be expensive. Thus, we also propose an extension of our method to effectively select a subsample of views to consider in the sampling. Computing the Z-order for each view results in a different partition of the point set into rank subsets, which we can interpret as different clusterings for the point set. Thus, we use the adjusted Rand Index [31] to measure the similarity between the clusterings of any two views. The rationale of this approach is that two views with similar distributions lead to similar Z-orders of the points, and thus only one plot is sufficient to reveal the correlations in the data. To obtain the final sample of views, we select the views with lowest pairwise similarities according to a farthest point sampling strategy. Then, we perform the sampling based only on the selected views, as illustrated in Figure 5.

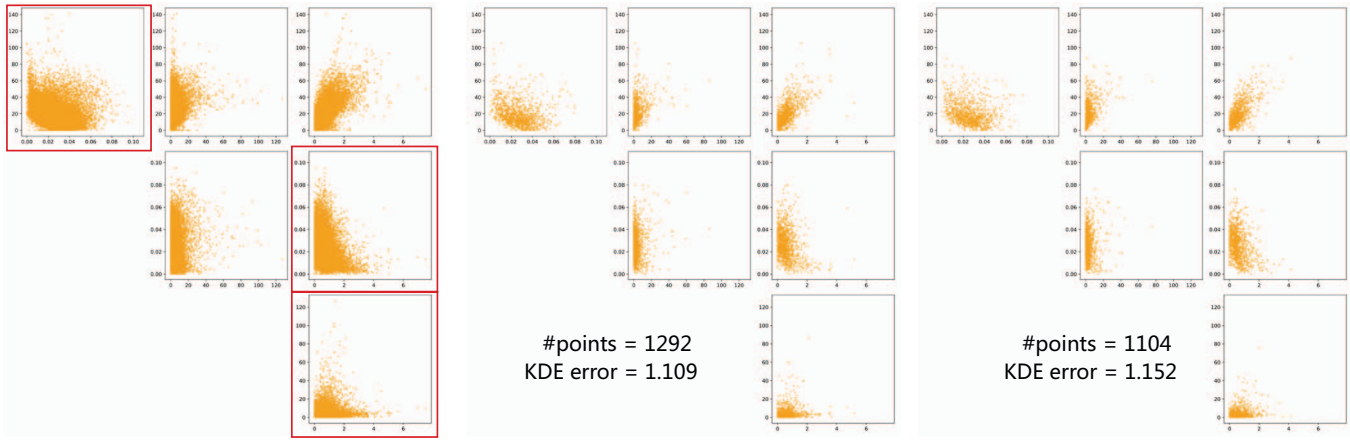
4 Results and Evaluations

In this section, we evaluate our method with quantitative and qualitative analyses, and also a user study, on a variety of datasets.

4.1 Datasets

For evaluating single-class scatterplot matrices, we test our method using the U.S. Air Pollution dataset[1] with 20K four-dimensional points, and one synthetic dataset with 5K three-dimensional points. For multi-class scatterplots, we use 15 synthetic datasets ($D_1 - D_{15}$) and 5 real

[1] https://www.kaggle.com/sogun3/uspollution



(a) Input with selected views (b) Sampling with all views (c) Sampling with selected views

Fig. 5. Example of view selection for sampling optimization. After comparing the clusterings of each view provided by the Z-order method, the three views marked with red squares in (a) are selected as representatives of the dataset. When sampling only with the selected views as in (c), the results are similar to sampling with all the views as in (b), both visually and in terms of approximation error. However, the sampling (c) can be computed in less time than (b) and requires less points.

datasets ($D_{16} - D_{20}$) that we selected, where D_{16} was provided by Zheng et al. [42], D_{17} is a Wine Quality dataset [12], and $D_{18} - D_{20}$ are subsets of the UCI HAR dataset [1]. For evaluating multi-class scatterplot matrices, we use 10 synthetic datasets which allow us to sample a large number of points. All the synthetic datasets were generated using the scikit-learn library in Python that randomly samples from random Gaussian distributions where we specify mainly the number of classes and points. The number of points in the datasets vary between 5K and 20K and the values in each dimension vary between -10 and 10.

4.2 Performance of our sampling method

In this section, we evaluate the performance of our method in terms of execution time and final sample size obtained.

Timing and size of output sample. Table 1 presents timing information for our method and statistics for a few selected datasets. Specifically, the number of views m is directly derived from the dimensionality d of the datasets according to $m = d \times (d-1)/2$, while the number of partitions depends on the number of views and classes. As we see from the table, our method is quite efficient and is able to sample 1K from 20K points in under 6 seconds for the most complex dataset with 6 classes. Although we set the target number of sample points to 1K, the final number of points can be more than the target, given that we need to cover all the subsets for all the views. Nevertheless, as we see from these examples, we require at most 1.5 times the target number of samples to cover all the sets. Note that the number of partitions that result from different Z-orders directly determines the complexity of the set cover problem, increasing the running time and the number of sampled points for a given dataset. These considerations demonstrate the necessity of view selection for sampling optimization, which can reduce the number of output points sampled.

Scalability with increasing views. To evaluate the scalability of the method when increasing the number of views in the dataset, we present an experiment where we sample 1K from 20K points while increasing the dimensionality of the points from 3 to 10. The number of views in then derived again by the relation $m = d \times (d-1)/2$. We show the results of this experiment in Figure 6, where we report the final number of sample points obtained and the running time required for sampling with increasing views. We observe that the number of sampled points remains less than 2.5 times the requested number, while the timing increases quadratically. However, despite the non-linear time complexity, for over 40 views and 20K points, the sampling still takes less than one minute.

Table 1. Time required for processing selected datasets and final number of sampled points, along with other dataset statistics. All the datasets consist of 20K points and our goal is to sample 1K points. Note that, for datasets with a single class, #partitions = #views, while for datasets with more than one class, #partitions = #views × 2.

Datasets				Time (s)		Result
#dim	#views	#classes	#partitions	Z-order	Set cover	#points
2	1	3	2	0.80	0.09	1083
2	1	6	2	0.79	0.10	1070
3	3	1	3	1.10	0.36	1103
3	3	3	6	1.25	1.04	1269
4	6	1	6	1.51	1.50	1291
4	6	6	12	1.99	3.67	1549

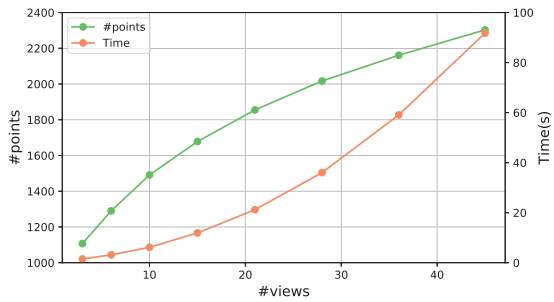


Fig. 6. Scalability of our method: number of sample points obtained and sampling time, while increasing the number of views in the dataset.

4.3 Evaluation of the quality of the sampling

To evaluate the overall quality of the samplings produced by our method, we compute the L_∞ KDE error defined in Eq. 3 to quantify the maximal differences between KDEs computed from the input point set and sampled set. We compare the KDE error obtained by our method to other methods. We perform quantitative evaluations mainly on single-class scatterplot matrices and individual multi-class scatterplots to show the direct effect of applying our method on these two types of visualizations. We show qualitative samples to illustrate these results. For multi-class scatterplots, we perform the evaluation on a larger variety

Table 2. Evaluation of our sampling method applied to different views V_i of the Air Pollution dataset, denoted $S(V_i)$. The numbers report the L_∞ KDE error of scatterplots in different views, denoted $E(V_i)$, where lower values are better.

Method	$E(V_1)$	$E(V_2)$	$E(V_3)$	$E(V_4)$	$E(V_5)$	$E(V_6)$
$S(V_1)$	**0.209**	0.00206	0.0108	1.135	5.419	0.0547
$S(V_2)$	0.219	**0.00205**	0.0108	1.247	5.654	0.0575
$S(V_3)$	0.217	0.00206	**0.0107**	1.168	5.885	0.0542
$S(V_4)$	0.212	0.00208	0.0110	**1.125**	5.417	0.0563
$S(V_5)$	0.213	0.00208	0.0109	1.175	**5.291**	0.0564
$S(V_6)$	0.215	0.00206	0.0107	1.187	5.499	**0.0533**
$S(V_{all})$	**0.204**	**0.00199**	**0.0105**	**1.134**	**5.224**	**0.0540**

Table 3. Evaluation of our sampling method applied to different views of the synthetic dataset. The metrics report the L_∞ KDE error of scatterplots in different views, where lower values are better.

Method	$E(V_1)$	$E(V_2)$	$E(V_3)$
$S(V_1)$	**0.255**	0.080	0.229
$S(V_2)$	0.264	**0.079**	0.250
$S(V_3)$	0.273	0.093	**0.206**
$S(V_{all})$	**0.205**	**0.100**	**0.206**

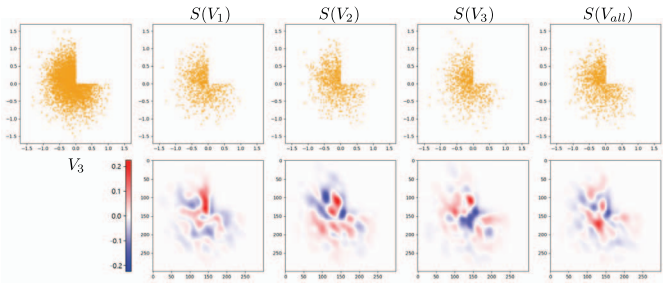


Fig. 7. Comparison of sampling results on view 3 (V_3) of the sythetic dataset, where the sampling was computed for specific and all views.

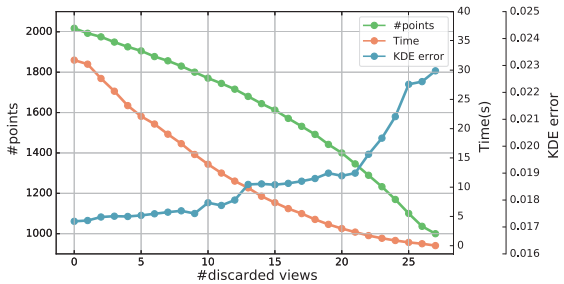


Fig. 8. Effect of view selection: number of sample points, sampling time, and KDE error, when increasing the number of discarded views.

of datasets and conduct user studies to compare with the latest methods such as blue noise sampling [9] and animation-based scatterplot visualization [10].

Evaluation on single-class scatterplot matrices. We compare the KDE errors among the following sampling methods: 1) Coresets computed for single views (different views lead to different coresets, thus providing different samplings); and 2) Our coreset sampled by considering all the views. For both methods, we sample both the 20K point Air Pollution dataset and the synthetic 5K point dataset to obtain 1K points. Note that this four-dimensional real dataset results in six different views and the three-dimensional synthetic dataset results in three different views in the corresponding scatterplot matrix, since we only consider all the variable combinations once.

Tables 2 and 3 show the KDE errors computed for each view when applying sampling methods 1 and 2 to the two datasets. In the tables, $S(V_i)$ denotes that we performed the sampling according to view V_i, while $E(V_i)$ means that we evaluated the KDE error according to V_i. $S(V_{\text{all}})$ denotes our method where we sample by considering all the views. Note that, since the sampling has a random component, we compute the sampling for each scatterplot 100 times and report the average KDE errors in the table. The results indicate that using the z-order method based on a specific view provides a low-error sampling result for that view, but the sampling quality for other views cannot be guaranteed. On the other hand, when using our method that considers all the views, we consistently obtain low-error results for each view. For some of the views of these two datasets, the error value when considering all the views even happens to be smaller than the error when applying the method to the specific view in question (e.g., compare rows 1 and 7 of column 1 in Table 2). This is likely due to the characteristics of these specific datasets.

As a qualitative sample of these results, Figure 7 shows the sampling results computed with the different methods and the corresponding KDE errors for view 3 of the synthetic dataset. In the figure, we see that the samplings obtained for view 3 or all the views provide a smaller KDE error than the sampling for other views.

View selection We also evaluate the gains obtained by performing view selection to discard uninformative views of the dataset. We sample 1K from 20K points in a dataset with feature dimension $d = 8$ and view number $m = 28$. We present this experiment in Figure 8, which shows the final number of sampled points, time required for sampling, and the KDE error with increasing number of discarded views. We see that the number of points and time clearly drop when the number of discarded views increases. On the other hand, the view selection also leads to the increase of the KDE error. Thus, the user needs to determine a trade-off when deciding the number of views to discard. In this example, discarding between 13 and 21 views provides a balanced trade-off between timing, sample size, and error.

Evaluation on multi-class scatterplots. We compare the KDE errors among the following sampling methods: 1) Random sampling (RS): we randomly sample n points from the dataset; 2) Multi-class blue noise sampling [9]; 3) Local view: we sample the coreset for each class separately and then combine the sets together; 4) Global view: we sample the coreset for the whole point set without considering the class labels; and 5) Local+global view: our coreset obtained by considering both per-class and global distributions. We use 15 synthetic datasets and 5 real datasets for the evaluation on multi-class scatterplots.

Table 4 shows the overall KDE errors using different methods for each dataset, where the overall KDE error is defined as the maximal value among the per-class KDE errors and global KDE error. Analyzing the table leads to the conclusion that our method that considers both the individual classes and the global view provides the lowest KDE errors in 19 of the 20 datasets, while the error is quite close to the top method for dataset D_8. Moreover, our method that considers only the local or global views also provides lower errors than the blue noise sampling for all the datasets, which is expected since the blue noise sampling does not optimize specifically for the KDE error.

Figure 9 shows more detailed statistics on the differences between the four methods for a few selected datasets. On the left of each subfigure, we show the scatterplot obtained with the entire input point set, where class colors were selected according to ColorBrewer [17]. On the right, we show the corresponding KDE errors for the global point distribution (denoted "V_{global}") as well as the maximal error for all the individual classes (denoted "V_{local}"). We see that our Z-order-based method always obtains lower KDE errors than blue noise sampling. The local-view-based method can provide good per-class samplings, but cannot guarantee a good global sampling. The global-view method provides results with the opposite effect. Finally, when considering

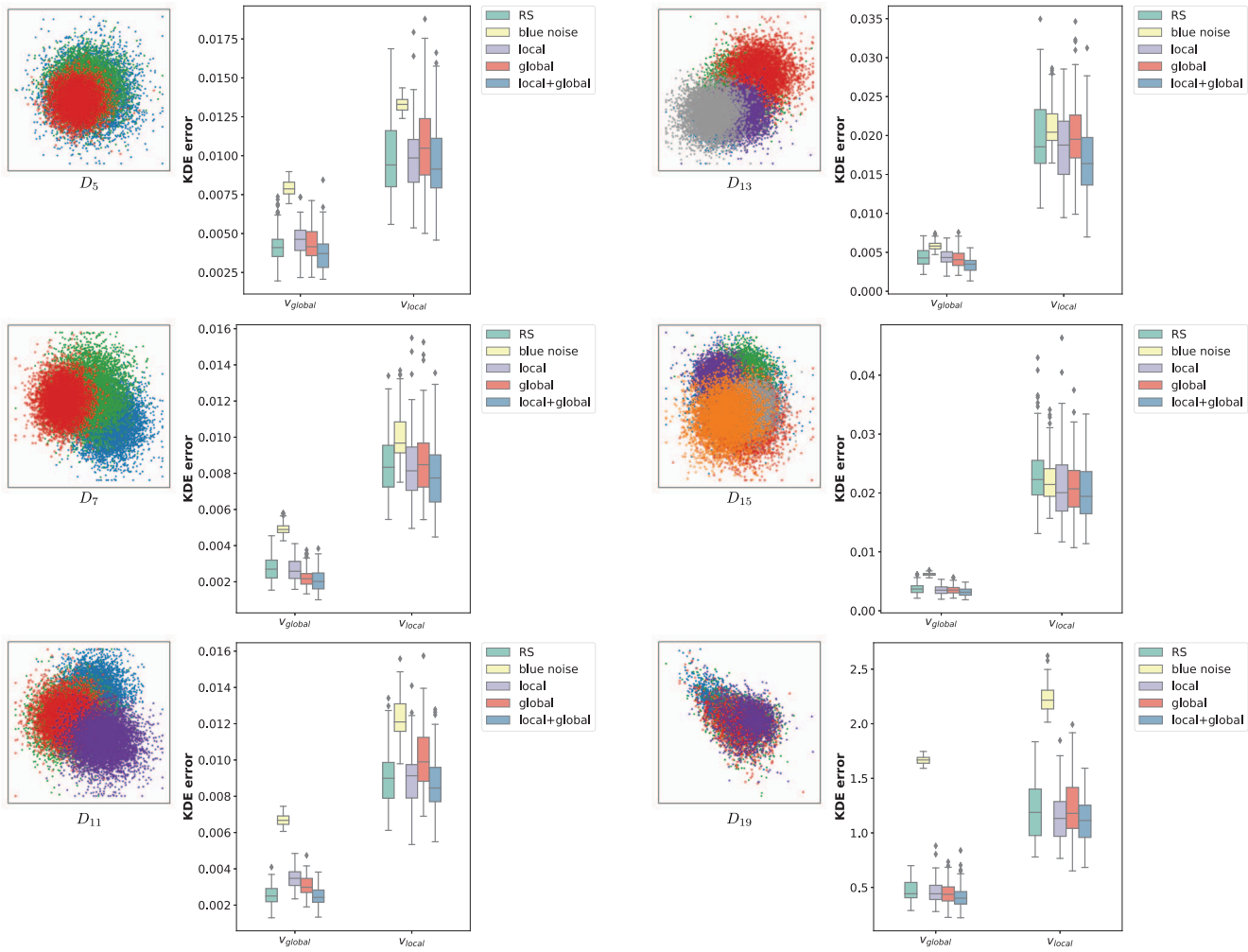


Fig. 9. Comparison of different sampling methods for multi-class scatterplots on six selected datasets. For each dataset, we show on the left the scatterplot drawn with the full point set as reference, while on the right the boxplots present the distribution of KDE errors, indicating the median, minimum, maximum, first and third quartiles, and outliers. We show one boxplot for each of the four sampling methods, both when considering the global distribution of points ("V_{global}"") and individual classes ("V_{local}""). Note that the lowest errors are consistently provided by our sampling method that considers the local and global views (blue boxplots), while all the Z-order-based samplings provide better results than the blue noise method.

both the local and global views together, we obtain consistently low errors for each class and for the global distribution.

4.4 User studies for task-based evaluation

In this section, we describe two user studies where we compared our method to existing methods for plotting multi-class scatterplots [9] and multi-class scatterplot matrices [10]. Note that there are no alternative methods specifically for plotting single-class scatterplot matrices, and thus we do not perform a comparative study for this case.

User study on multi-class scatterplots. We conduct a user study composed of four tasks. Since the state-of-the-art method for sampling multi-class scatterplots is the blue noise sampling of Chen et al. [9], we use the two tasks from Chen et al.'s user study to evaluate our method. We complement these tasks with two new tasks where we evaluate the global density of points and the global perceptual similarity to the original point distribution, to show that our method also improves the scatterplots in these two aspects. The tasks involve analyzing the results of our sampling and the results of the blue noise sampling method, and a few of the tasks also involve comparing the results to a conventional scatterplot drawn without any optimization. As follows, we briefly describe the four tasks and then present our results:

- *Task 1 (T1): Data class identification.* Participants had to identify the number of classes in a marked area, where the ground-truth can be extracted from the point set. We gave a score of 1 for correct answers and 0 to incorrect answers.
- *Task 2 (T2): Relative per-class density recognition.* Participants had to choose the answer that best describes the relative density order of different classes in a marked region. The ground-truth for the density of each class can also be extracted from the point set, and we gave a score of 1 if the user selected the correct density order from a list, or 0 otherwise.
- *Task 3 (T3): Relative global density recognition.* Participants had to compare the densities of two marked regions and select the region with higher density. We asked the users to do the comparison without considering the different labels, i.e., compare the global density. The ground-truth density can also be extracted from the point set, and we gave a score of 1 for correct answers and 0 to incorrect answers.
- *Task 4 (T4): Perceptual similarity.* Participants had to choose which sampling result is perceptually more similar to the original point distribution. Users had to choose a single scatterplot out of a set of two plots, which included the blue noise sampling result and our sampling result. The answers provide the preference of the user when comparing a scatterplot to the original plot.

Table 4. Comparison of our sampling method against other sampling methods on multi-class scatterplots. The numbers report the overall KDE error for the different methods (columns), where lower values are better. Please refer to the text for an explanation of each method.

ID	#classes	RS	blue noise	local	global	local + global
D_1	3	0.0146	0.0125	0.0121	0.0120	**0.0117**
D_2	3	0.0188	0.0189	0.0184	0.0187	**0.0171**
D_3	3	0.0242	0.0265	0.0234	0.0236	**0.0218**
D_4	3	0.0328	0.0436	0.0336	0.0316	**0.0308**
D_5	3	0.00997	0.0133	0.00989	0.0102	**0.00964**
D_6	3	0.0108	0.0135	0.0105	0.0106	**0.0103**
D_7	3	0.00851	0.0100	0.00850	0.00869	**0.00837**
D_8	3	0.0136	0.0136	**0.0131**	0.0135	0.0133
D_9	3	0.00992	0.0101	0.00940	0.00966	**0.00934**
D_{10}	3	0.00986	0.00998	0.00977	0.00986	**0.00913**
D_{11}	4	0.00904	0.0124	0.00897	0.00904	**0.00871**
D_{12}	4	0.0145	0.0151	0.0143	0.0142	**0.0134**
D_{13}	5	0.0201	0.0211	0.0199	0.0201	**0.0189**
D_{14}	5	0.0270	0.0271	0.0253	0.0269	**0.0243**
D_{15}	6	0.0230	0.0218	0.0213	0.0212	**0.0207**
D_{16}	2	0.0	0.0	0.0	0.0	0.0
D_{17}	2	2.757	2.875	2.752	2.776	**2.415**
D_{18}	3	1.356	2.699	1.325	1.337	**1.252**
D_{19}	4	1.201	2.236	1.158	1.230	**1.133**
D_{20}	5	1.414	2.821	1.341	1.395	**1.338**

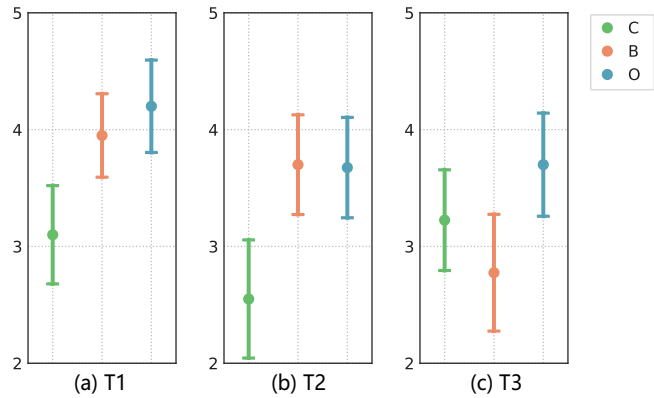


Fig. 10. Results of the user study on multi-class scatterplots. The boxplots show the average (higher is better) and standard deviation of the scores obtained by users for each sampling method on tasks T1, T2 and T3, where C, B, and O represent conventional scatterplot, blue noise sampling, and our sampling, respectively. Note the higher scores obtained with our sampling on T1 and T3, and comparative result on T2.

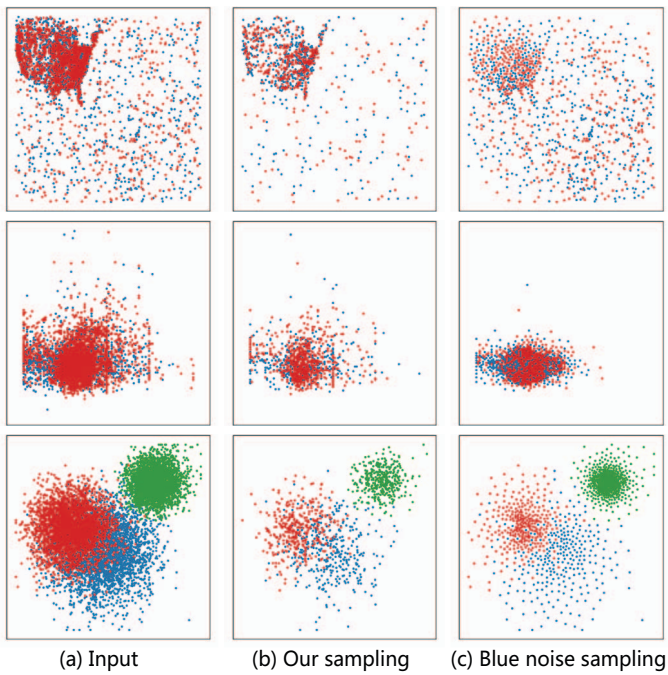


Fig. 11. Three example questions of T4 in the user study on multi-class scatterplots, where the users selected our sampling as being more perceptually similar to the original point distribution. Note how the results of blue noise sampling tend to be too uniformly distributed.

We recruited 40 participants for this study, where 32 were male and 8 were female, 30 were graduate students and 10 were undergraduate students. Their ages ranged from 18 to 27 years old. All the participants reported that they did not have any known form of color blindness. For T1, T2 and T3, each participant had to answer 15 questions, including 5 questions for each type of point distribution, i.e., the original distribution, the blue noise sampling, and our sampling. The marked regions shown for each participant were obtained from different datasets to avoid interference. For T4, each participant had to answer 8 questions. The scatterplots were randomly permuted before being presented to the user, to avoid any kind of bias in the selection. Our sampling was performed without outlier inclusion.

Figure 10 shows the overall user performances on T1, T2 and T3. We can see that with the sampling provided by our method, the users obtained better scores for T1 and T3, and scores comparable to blue noise sampling for T2. Especially for T3, we see that users had an increase of 33.3% in the performance when using our sampling. For T4, the users chose our sampling as being perceptually more similar to the original point distribution in 60.3% of the answers. The main reason given by the participants for this preference is that our method better preserves the original data distribution while the blue noise method tends to sample evenly-distributed point sets, which look artificial according to the users. Figure 11 shows three example questions for T4 where users selected our sampling as better representing the original point distribution.

User study on multi-class scatterplot matrices. We performed a user study to compare the animation-based method of Chen et al. [10] with our methods for sampling scatterplots and views of scatterplot matrices. We used the same two tasks introduced by Chen et al., which we name T5 and T6 in our paper. Compared to the previous study where participants had to answer questions related to marked regions, in the following tasks, the users had to select the regions that satisfy the given requirements by dragging a fixed-size rectangle on the screen.

- *Task 5 (T5): Targeted density identification.* Participants had to identify the region in a specific cell/view with the most circles/points.
- *Task 6 (T6): Diverse density identification.* Participants had to identify the region in any cell/view with the most circles/points and at least four different classes.

Note that one specific view is given in T5, while the entire scatterplot matrix is given in T6. We recruited 30 participants for this study. Of these participants, 24 were male and 6 were female, 21 were graduate students and 9 were undergraduates. Their ages ranged from 18 to 27 years old. All the participants reported that they did not have any known form of color blindness. For both T5 and T6, each participant had to answer 15 questions including 5 questions for each type of point distribution, i.e., original static distribution, the one with animation, and our static sampling result. The scatterplots or scatterplot matrices were randomly permuted before being presented to the user, to avoid any kind of bias in the selection. To evaluate a user's performance, we computed the number of points in the selected regions and kept track of the time taken to complete the tasks. Similarly to Chen et al.'s study [10], after completing both tasks, participants were asked whether

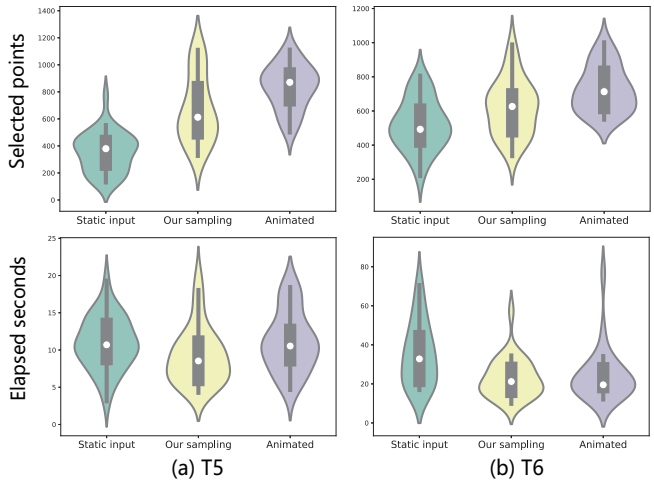


Fig. 12. Results for the user study on multi-class scatterplot matrices. We use the same visualization as Chen et al. [10], where the violin plots show the density of ratings, the boxplots show the first and third quartiles, the whiskers show 1.5 times the inter-quartile range, and the white circle shows the median.

they found each technique easy to interpret, and which technique they preferred and why.

Figure 12 shows the overall performances on tasks T5 and T6. From analyzing the results, we see that with our sampling results, the performance of the users is comparable to the animation-based method, although requiring significantly less data. We also see that our sampling results provide better performance than the original full point set. For the question about preference, 60% of users chose our method, 23.3% chose the animation-based method, 10% indicated that the methods are equivalent, and 6.7% chose the option "hard to determine". The main reasons that the participants gave for preferring our method over others were: 1) Using a sampling better represents the data distribution, allowing the users to quickly identify the data density. On the other hand, although the animation-based method uses flickering to indicate the density, the data itself is still quite dense and it is difficult to see its details; 2) For regions in the scatterplot that do not have overdrawing of points from different classes, the animation-based method that keeps drawing points with the same color can make it hard to understand the data distribution, while our static sampling can show the data more clearly; and 3) The animated plots are sometimes too distracting, which may cause the users to ignore important information.

5 Conclusion

We introduced a method for simultaneously optimizing the sampling of a point set for a collection of views or classes. Our method is based on the idea of partitioning the input set into subsets of points, according to Z-order curves computed for the multiple views or classes, and then obtaining a point sampling that covers all the subsets via set cover optimization. We showed experimentally that, in this manner, the sampling has lower KDE approximation errors for all the views or classes considered when comparing to other methods. We confirmed with a quantitative evaluation that our method provides accurate approximations of a variety of datasets. Moreover, we demonstrated with two studies that the performance of users in a variety of scatterplot analysis tasks is either comparable or superior to their performance with existing methods such as blue noise sampling and animation-based multi-class SPLOM visualization, especially enabling users to better distinguish the density of different regions of the plots or overlap of classes.

Limitations and future work. In practice, our method cannot guarantee that the final plots do not suffer from overdraw. The sampling of coresets does not directly optimize for reducing this effect, which is directly considered by the blue-noise sampling method. Moreover, our method does not generate sets with the exact number of samples n required by the users, due to the approximation method we used, although this is also a limitation for alternatives such as the blue-noise sampling method. In addition, we limited our evaluation to experiments mainly with datasets generated from Gaussian distributions, due to the simplicity of synthesizing them. However, our method can be applied to other types of distributions, as shown in the results for real datasets.

Our method is orthogonal to many classic methods that seek to reduce overdraw problems in scatterplots, such as the use of translucent markers or color blending, combined with a selection of adequate color palettes for drawing multiple classes. Thus, it can be readily combined with these techniques to provide a more complete solution to mitigate overdraw problems. Although we limited our experiments to comparing our method to alternative sampling methods and an animation-based drawing method, future work could also evaluate the coupling of our sampling method with classic overdraw-reduction techniques to reveal the most effective combination for scatterplot visualization. In addition, although we showed that users preferred our static sampling over animated plots, the two methods can also be combined when suitable, which could lead to better visualizations for certain types of data. Specifically, the use of our sampling can overcome problems such as continuous drawing of points of the same class in the animation, which some users found confusing.

In relation to the parameters of our method, currently the subsample size n has to be manually specified by the users. However, we foresee that it should be possible to develop an algorithm that automatically determines n based on the resolution of the display/image where the scatterplot is being drawn. This can be framed as the design of a method that explicitly reduces the overdraw for each pixel of the output. Moreover, future user studies could also evaluate the impact that the selection of n has on the interpretation of the plots.

Finally, we believe that our sampling method could also be used with other techniques for visualizing massive datasets with single/multiple classes. Thus, another direction for future work is to investigate contexts where the sampling can be applied. A more immediate direction for future work is to investigate the use of our sampling method with more complex data types such as structured or graph-based data, which can be addressed with the use of special types of KDE kernels.

Acknowledgments

We thank the reviewers for their valuable comments. This work was supported in parts by NSFC (61872250, 61602311), GD Science and Technology Program (2015A030312015), GD Leading Talents Program (00201509), Shenzhen Innovation Program (JCYJ20170302153208613, KQJSCX20170727101233642), LHTD (20170003), NSERC (2015-05407), DFG (422037984), and the National Engineering Laboratory for Big Data System Computing Technology.

References

[1] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *Esann*, 2013.

[2] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 14(6):1428–1435, 2008. doi: 10.1109/TVCG.2008.119

[3] M. Behrisch, M. Blumenschein, N. W. Kim, L. Shao, M. El-Assady, J. Fuchs, D. Seebacher, A. Diehl, U. Brandes, H. Pfister, et al. Quality metrics for information visualization. *Computer Graphics Forum*, 37(3):625–662, 2018.

[4] E. Bertini and G. Santucci. By chance is not enough: preserving relative density through nonuniform sampling. In *Proc. Int. Conf. on Information Visualisation*, pp. 622–629. IEEE, 2004. doi: 10.1109/IV.2004.1320207

[5] E. Bertini and G. Santucci. Quality metrics for 2d scatterplot graphics: automatically reducing visual clutter. In *International Symposium on Smart Graphics*, pp. 77–89. Springer, 2004. doi: 10.1007/978-3-540-24678-7_8

[6] E. Bertini and G. Santucci. Give chance a chance: modeling density to enhance scatter plot quality through random data sampling. *Information Visualization*, 5(2):95–110, 2006. doi: 10.1057/palgrave.ivs.9500122

[7] E. Bertini, A. Tatu, and D. Keim. Quality metrics in high-dimensional data visualization: An overview and systematization. *IEEE Trans. Vis.*

& Comp. Graphics, 17(12):2203–2212, 2011. doi: 10.1109/TVCG.2011.229

[8] D. B. Carr, R. J. Littlefield, W. Nicholson, and J. Littlefield. Scatterplot matrix techniques for large n. *Journal of the American Statistical Association*, 82(398):424–436, 1987. doi: 10.2307/2289444

[9] H. Chen, W. Chen, H. Mei, Z. Liu, K. Zhou, W. Chen, W. Gu, and K.-L. Ma. Visual abstraction and exploration of multi-class scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 20(12):1683–1692, 2014. doi: 10.1109/TVCG.2014.2346594

[10] H. Chen, S. Engle, A. Joshi, E. D. Ragan, B. F. Yuksel, and L. Harrison. Using animation to alleviate overdraw in multiclass scatterplot matrices. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, p. 417. ACM, 2018. doi: 10.1145/3173574.3173991

[11] V. Chvatal. A greedy heuristic for the set-covering problem. *Mathematics of Operations Research*, 4(3):233–235, 1979.

[12] P. Cortez, A. Cerdeira, F. Almeida, T. Matos, and J. Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision Support Systems*, 47(4):547–553, 2009.

[13] 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*, pp. 167–176. ACM, 2002. doi: 10.1145/1556262.1556289

[14] G. Ellis and A. Dix. Density control through random sampling: an architectural perspective. In *International Conference on Information Visualisation*, 2002. doi: 10.1109/IV.2002.1028760

[15] G. Ellis and A. Dix. A taxonomy of clutter reduction for information visualisation. *IEEE Trans. Vis. & Comp. Graphics*, 13(6):1216–1223, 2007. doi: 10.1109/TVCG.2007.70535

[16] D. Feng, L. Kwock, Y. Lee, R. M. Taylor, et al. Matching visual saliency to confidence in plots of uncertain data. *IEEE Trans. Vis. & Comp. Graphics*, 16(6):980, 2010. doi: 10.1109/TVCG.2010.176

[17] M. Harrower and C. A. Brewer. ColorBrewer.org: an online tool for selecting colour schemes for maps. *The Cartographic Journal*, 40(1):27–37, 2003. doi: 10.4324/9781351191234-18

[18] R. M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pp. 85–103. Springer, 1972.

[19] D. A. Keim, M. C. Hao, U. Dayal, H. Janetzko, and P. Bak. Generalized scatter plots. *Information Visualization*, 9(4):301–311, 2010. doi: 10.1057/ivs.2009.34

[20] D. A. Keim and A. Herrmann. The gridfit algorithm: An efficient and effective approach to visualizing large amounts of spatial data. In *Proc. IEEE Conf. on Visualization*, pp. 181–188. IEEE, 1998. doi: 10.1109/VISUAL.1998.745301

[21] E. M. Knox and R. T. Ng. Algorithms for mining distancebased outliers in large datasets. In *Proceedings of the international conference on very large data bases*, pp. 392–403. Citeseer, 1998.

[22] M. Krzywinski and B. Wong. Points of view: plotting symbols. *Nat Methods*, 10(6), 2013. doi: 10.1038/nmeth.2490

[23] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. Springer Science & Business Media, 2007. doi: 10.1007/978-0-387-39351-3

[24] J. Li, J.-B. Martens, and J. J. van Wijk. A model of symbol size discrimination in scatterplots. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 2553–2562, 2010. doi: 10.1145/1753326.1753714

[25] M. Luboschik, A. Radloff, and H. Schumann. A new weaving technique for handling overlapping regions. In *Proc. Int. Conf. on Advanced Visual Interfaces*, pp. 25–32. ACM, 2010. doi: 10.1145/1842993.1842999

[26] J. Matejka, F. Anderson, and G. Fitzmaurice. Dynamic opacity optimization for scatter plots. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pp. 2707–2710. ACM, 2015. doi: 10.1145/2702123.2702585

[27] A. Mayorga and M. Gleicher. Splatterplots: Overcoming overdraw in scatter plots. *IEEE Trans. Vis. & Comp. Graphics*, 19(9):1526–1538, 2013. doi: 10.1109/TVCG.2013.65

[28] L. Micallef, G. Palmas, A. Oulasvirta, and T. Weinkauf. Towards perceptual optimization of the visual design of scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 23(6):1588–1599, 2017. doi: 10.1109/TVCG.2017.2674978

[29] M. Novotny and H. Hauser. Outlier-preserving focus+ context visualization in parallel coordinates. *IEEE Trans. Vis. & Comp. Graphics*, 12(5):893–900, 2006. doi: 10.1109/TVCG.2006.170

[30] J. M. Phillips and W. M. Tai. Near-optimal coresets of kernel density estimates. In *Symposium on Computational Geometry*, 2018.

[31] W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

[32] M. Sedlmair, T. Munzner, and M. Tory. Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE Trans. Vis. & Comp. Graphics*, 19(12):2634–2643, 2013. doi: 10.1109/tvcg.2013.153

[33] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. CRC Press, 1986.

[34] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan. Selecting good views of high-dimensional data using class consistency. *Computer Graphics Forum*, 28(3):831–838, 2009. doi: 10.1111/j.1467-8659.2009.01467.x

[35] G.-M. P. TUFTE E. R. *The visual display of quantitative information*. Graphics Press, 1983.

[36] A. Unwin, M. Theus, and H. Hofmann. *Graphics of large datasets: visualizing a million*. Springer Science & Business Media, 2006. doi: 10.1007/0-387-37977-0

[37] V. V. VAZIRANI. *Approximation Algorithms*. Springer, 2001.

[38] Y. Wang, X. Chen, T. Ge, C. Bao, M. Sedlmair, C.-W. Fu, O. Deussen, and B. Chen. Optimizing color assignment for perception of class separability in multiclass scatterplots. *IEEE Trans. Vis. & Comp. Graphics*, 25(1):820–829, 2019. doi: 10.1109/TVCG.2018.2864912

[39] L.-Y. Wei. Multi-class blue noise sampling. *ACM Trans. Graph. (SIGGRAPH)*, 29(4):79, 2010. doi: 10.1145/1778765.1778816

[40] L. Wilkinson. *The grammar of graphics*. Springer Science & Business Media, 2006. doi: 10.1007/0-387-28695-0

[41] A. Woodruff, J. Landay, and M. Stonebraker. Constant density visualizations of non-uniform distributions of data. In *Proc. ACM Symposium on User Interface Software and Technology*, pp. 19–28, 1998. doi: 10.1145/288392.288397

[42] Y. Zheng, J. Jestes, J. M. Phillips, and F. Li. Quality and efficiency for kernel density estimates in large data. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, pp. 433–444. ACM, 2013.

[43] Y. Zheng, Y. Ou, A. Lex, and J. M. Phillips. Visualization of big spatial data using coresets for kernel density estimates. In *2017 IEEE Visualization in Data Science (VDS)*, pp. 23–30. IEEE, 2017.