A Recursive Subdivision Technique for Sampling Multi-class Scatterplots

Xin Chen, Tong Ge, Jian Zhang, Baoquan Chen,
Chi-Wing Fu, Oliver Deussen and Yunhai Wang

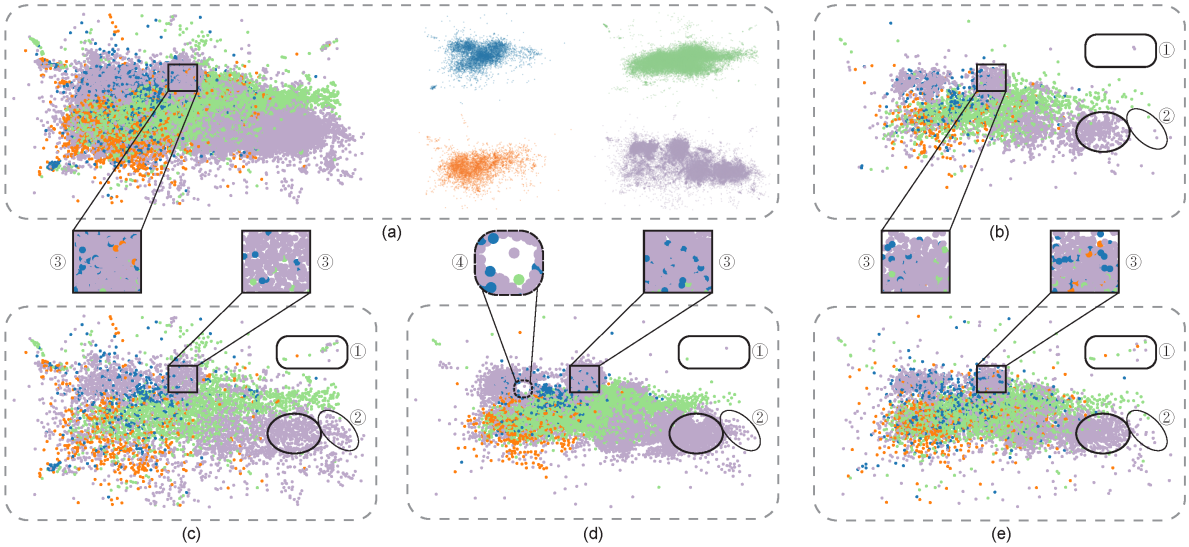


Fig. 1. Different sampling methods for presenting the four-class *Person Activity* data [8]. (a) The left shows the input scatterplots with 100K points and the right shows the four classes separately, where the patterns of each class are obscured in the main plot, e.g., the three sub-clusters in the purple class, due to overdraw. We re-sample the data into ∼5000 points using (b) random sampling, (c) non-uniform sampling [4], (d) multi-class blue noise sampling [11], and (e) our method. The results show that our method better preserves major outliers (see the rounded boxes labeled with "1"), relative data densities (see the ellipse labeled with "2" to compare (c) with (d)), and the relative class densities (see the orange points shown in the squares labeled with "3" in (a)-(e)), without introducing obvious visual artifacts such as highlighted by the square in (d) labeled with "4". Points for all results are rendered in random order.

Abstract—We present a non-uniform recursive sampling technique for multi-class scatterplots, with the specific goal of faithfully presenting relative data and class densities, while preserving major outliers in the plots. Our technique is based on a customized binary kd-tree, in which leaf nodes are created by recursively subdividing the underlying multi-class density map. By backtracking, we merge leaf nodes until they encompass points of all classes for our subsequently applied outlier-aware multi-class sampling strategy. A quantitative evaluation shows that our approach can better preserve outliers and at the same time relative densities in multi-class scatterplots compared to the previous approaches, several case studies demonstrate the effectiveness of our approach in exploring complex and real world data.

Index Terms—Scatterplot, multi-class sampling, kd-tree, outlier, relative density

1 Introduction

Scatterplots are widely used for visualizing pairwise relationships between quantitative variables. By encoding data points as visual marks (e.g., dots), they effectively show the correlations among variables, data clusters, and outliers, as well as other data patterns [37]. Multi-class scatterplots are effective in visualizing labeled data by color-coding visual marks based on class labels. Such plots are also good for visualizing 2D data [40] generated by means of dimensionality reduction [29].

Scatterplots, however, often suffer from the overdraw problem, i.e., overlapping visual marks in high-density regions. As an example see the purple class on the right of Fig. 1(a), which consists of three sub-groups, but the separation between these sub-groups is obscured in the plot that shows all the classes (left of Fig. 1(a)). The most straightforward way to alleviate this problem is to modulate the appearance of the marks, e.g., reducing their size and making them semi-transparent. This, however, will not work in cases with severe overdraw. Another simple solution is to show a separate plot for each class (right of Fig. 1(a)). However, this approach cannot represent how classes correlate with one another and demands for much more screen space for showing the details (e.g., outliers) in the classes. A number of other approaches [16] have been proposed to solve the problem, among which density estimation and sampling are two commonly-used ones.

- *X. Chen, T. Ge and Y. Wang are with Shandong University. Email: {cloudseawang, chenxin199634, getong95}@gmail.com.*
- *B. Chen is with Peking University. E-mail: baoquan.chen@gmail.com.*
- *C.-W. Fu is with the Chinese University of Hong Kong and Guangdong Prov. Key Lab. of CV and VR Tech., SIAT. E-mail: cwfu@cse.cuhk.edu.hk.*
- *J. Zhang is with CNIC, CAS. E-mail: zhangjian@sccas.cn.*
- *O. Deussen is with Konstanz University, Germany and Shenzhen VisuCA Key Lab, SIAT, China. E-mail: oliver.deussen@uni-konstanz.de.*
- *X. Chen and T. Ge are joint first authors.*
- *Y. Wang and C.-W. Fu are the co-corresponding authors.*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

Density estimation computes a continuous density function for a given scatterplot and displays this function in a color-coded way instead of the data points. The approach is effective for showing patterns in high-density regions, but it often misses outliers at low-density areas. For multi-class scatterplots the density fields of different classes are blended by this approach, which might yield to misleading visual representations in overlapping regions due to color blending. Although it is possible to alleviate these issues by highlighting dense regions with smooth contours [35], the approach might introduce additional clutter due to multiple contours, especially for data with many classes.

Another approach is sampling, which selects a subset of data points for display to reduce overdraw and visual clutter. Since selecting points by simple random sampling will void out data patterns in low-density areas (see Fig. 1(b)), several non-uniform sampling methods have been proposed. Bertini and Santucci [4] developed a perception-driven method [6] for preserving the relative density difference between different regions. This method, however, might not be able to preserve relative data densities for high-density regions (see the ellipse in Fig. 1(c)), and since it was designed for single-class scatterplots, it also not able to preserve class features in multi-class scatterplots. For example, the orange points in the square with label "3" in Figs. 1(a) and (e) are not shown in subfigure (c). More recently, Chen et al. [11] leveraged a multi-class blue noise sampling method [45] for multi-class scatterplots. This method preserves relative data densities and class densities to a certain extent, but it might not be able to preserve outliers and rare classes. For example, the rounded box labeled with "1" in Fig. 1(d) has few points and the square with label "3" does not contain orange points. Moreover, this method likely produces visual artifacts [17] (see the round dashed box with label "4" in Fig. 1(d)).[1]

In this paper, we present a non-uniform recursive sampling method for multi-class scatterplots. Our approach is based on four requirements for exploring multi-class scatterplots [39]: (i) keep major outliers in the scatterplots; (ii) preserve relative data densities, i.e., the data densities between two regions; (iii) maintain relative densities between classes, i.e., more faithfully representing the densities of different classes for each region; and (iv) minimize visual artifacts. Like other works, non-uniform sampling [4] is adopted to meet the first two requirements. While multi-class blue noise sampling [11] meets the second and third requirements to some extent, as far as we know, all requirements have not been fully adopted in any existing work on sampling for visualizing multi-class scatterplots. Fig. 1 compares the results produced by different methods on the Person Activity data [8]. Only our results (see Fig. 1(d)) meet all the four requirements.

Our approach is based on a recursive subdivision that hierarchically partitions a given multi-class scatterplot into nonempty regions, modeled as a customized 2D kd-tree. This subdivision is guided by the sampling ratio of each two sibling nodes that describes how much data is represented by the associated leaf nodes. If the sampling ratio of one node is smaller than the other or only slightly larger, it should be further partitioned so as to preserve relative data densities. By doing so, low-density regions are maintained while their density difference to regions of high-density is preserved, see an example in Fig. 3(c).

Once such a tree-based partition was created, randomly sampling a single data point from each region would allow us to preserve relative data densities between the different regions. On the other hand, the relative class densities inside the regions might not be preserved, since some classes with low densities might be lost. To address this issue, we perform backtracking from each leaf node to find an internal node whose number of visual points is large enough to preserve relative class densities. Doing so, we represent each class with a single region at least and classes with more data samples still have a larger chance to be represented by more random samples.

We evaluated our approach using 37 multiclass scatterplots [8] and quantitatively compared the quality of our results with previous methods. For the tested datasets, our method is capable of producing results with a better preservation of relative data density, relative class density and outliers. In addition, we present an extension to multi-dimensional data and show some sampling results on multi-class scatterplot matrices. In summary, the main contributions of this paper are:

- We propose a recursive subdivision based sampling approach, which is based on a tree-based partition and a sampling approach that aims at preserving relative data densities and relative class densities (Section 4), and
- We quantitatively evaluate the sampling results of our method and conduct three case studies that show the usefulness of our approach (Section 5).

[1]To avoid any bias on specific classes, points in all the results of Fig. 1 are rendered in a random order.

2 Related Work

2.1 Overdraw Reduction for Scatterplots

Overdraw is one of the main challenges for scatterplots. Various methods have been proposed to relieve it by altering the marker size, color, transparency, position, density, or by animation [12]. The majority of existing methods can be classified into four categories: appearance optimization, data jittering, density estimation, and visual sampling.

Appearance optimization is an intuitive approach to relieve overdraw, e.g., reducing marker sizes [30, 47], changing marker shapes [27], and making markers semi-transparent [46]. The first two strategies are suitable for moderate overdraw but cannot deal with the situation that markers are already very small. Setting a proper opacity to make markers semi-transparent is more scalable. Matejka et al. [34] presented a user-driven model for automatically setting the opacity based on crowdsourcing studies. Recently, Micallef et al. [36] pushed this line further by simultaneously optimizing marker opacity, size, and aspect ratio of a plot based on some visual quality metrics. When all these variables cannot help, Dang et al. [13] suggested to stack visual markers in an additional dimension based on the density. This strategy does not, however, scale for large data.

While all the above methods focus on single-class settings, Luboschik et al. [33] introduced a weaving technique to present overlapping regions in multi-class scatterplots, where each class can be easily identified by its distinct hue. Recently, Wang et al. [44] proposed a data-aware method to automatically find the best color assignment scheme to improve the perception of class separability. All these strategies are orthogonal to our sampling method, and can be combined with it to further reduce overdraw and to enhance user perception.

Data jittering relieves overdraw by slightly displacing the positions of overlapping markers to reveal them. It has been used in commercial systems like Spotfire [1]. Keim and Herrmann [26] proposed a space-filling pixel-based technique to shift overplotted data points along predefined curves to the nearest unoccupied pixels. Since a large jittering might introduce non-existent fake patterns, Keim et al. [25] developed generalized scatterplots, where users can control the overdraw and distortion via a non-linear warping scheme. In general, data jittering should be used with care, and cannot handle cases of extreme overdraw.

Density estimation is an alternative approach [10, 43] that shows the point data as a color-coded density plot, or as a set of contour lines. Hence, dense regions can be better characterized, but outliers and sparse regions might be missed. Bachthaler and Weiskopf [2] created a continuous density field by using respective interpolation schemes for the data defined on continuous domains. Novotny and Helwig [38] converted data into a density-based representation by using multi-dimensional binning and specially treated sparse data regions to preserve trends and outliers. Feng et al. [18] generated density plots using kernel density estimation [41] and suggested to combine mean emphasis with density plots to highlight outliers. Mayorga and Gleicher [35] introduced splatterplots to explicitly show outliers as discrete markers and dense regions as smooth contours. However, it is quite challenging to explore these visualizations for multi-class color-coded density fields, especially for overlapping regions due to color blending. Our sampling method also considers multi-class density information and is able to show classes and outliers while preserving relative class densities.

Visual sampling is another common approach to avoid overdraw [16]. Ellis and Dix [14, 15] used random sampling to visually reduce data

density. To determine the right sampling ratio, Bertini and Santucci [5] modeled the relationship between the visual density and clutter, and presented an automatic method to preserve the relative densities. Later, they introduced a non-uniform sampling method [4] for preserving low-density areas. This method, however, might fail to represent relative densities in high-density regions, since such regions might be occupied by the same number of distinct pixels. Moreover, it cannot deal with multi-class scatterplots. Instead, Chen et al. [11] converted a scatterplot into a density field by kernel density estimation and applied multi-class blue noise sampling [45] to reduce the overdraw. This approach might not produce reasonable results, if the density field does not characterize the data well. Furthermore, blue noise sampling could introduce local patterns (typically hexagonal sub-arrangements of the dots) that do not exist in the actual data points. Also, using such density fields to control the minimum spacing between point samples might create additional artifacts (see the holes highlighted in Fig. 1(d) rectangle 4), due to the conflict between multi-class samples. Moreover, point selection requires checking on all un-sampled points, while conflict checking requires visiting all sampled points, thus resulting in an expensive search that limits its applicability to large data sets encountered in practice. In this work, we take a different approach to work directly on the discrete input data points in the given scatterplots and explicitly consider the relative class density to faithfully guide the re-sampling.

2.2 Quantitative Metrics for Scatterplots

The automatic evaluation of visualization quality is a fundamental topic in information visualization. Bertini et al. [7] and Behrisch et al. [3] systematically surveyed various quality metrics. Here, we mainly focus on those designed for measuring the visual clutter in scatterplots.

The first attempt for defining metrics for visualizations came from Tufte [42], who proposed a set of measures to assess the effectiveness of paper-based visualizations. Among them, *data density* refers to the ratio between the number of displayed data samples and the corresponding area in the data graphics. To reduce the visual clutter in interactive visualizations, Bertini and Santucci [5] proposed a quality metric to compare the visible data density in image space relative to the data density in data space, and used the metric to find an optimal sampling factor. Later, they propose a metric [6] to compute low-density areas removed by sampling. In this paper, we use these two metrics to quantify the preservation of relative data density and outliers of our approach. We further extend them for quantifying the preservation of relative class density and outliers (rare classes) in multi-class scatterplots.

3 Formal Definitions

In this section, we formally define basic components used in the design of our approach. First, we represent a given multi-class scatterplot of m classes as a multi-class *data density* map D: $\mathbb{R}^2 \to \mathbb{R}^{m+1}$, where $D^i(\mathbf{x})(1 \le i \le m)$ is the density of the i-th class at pixel region $\mathbf{x}$ in the density map, $D^0(\mathbf{x}) = \sum_{i=1}^{m} D^i(\mathbf{x})$ is the density sum over all the classes, and the density is measured by counting the number of points in $\mathbf{x}$. Likewise, the density of points in a region $\Omega \subset \mathbb{R}^2$ on the density map is defined as $D^i(\Omega) = \sum_{\mathbf{x}\in\Omega} D^i(\mathbf{x})$. Note that we use D^i and D^0 to denote the density map of the i-th class and the whole data, respectively. See Fig. 3(a) for a typical example of D^i for a two-class scatterplot.

Our sampling method selects one *data sample* per pixel region. A *point sample* is referred to as a data sample that is to be placed in a pixel region in the output visualization. Similar to the density map, we define a multi-class *visual density* map D_v: $\mathbb{R}^2 \to \mathbb{R}^{m+1}$, where $D_v^i(\mathbf{x})(1 \le i \le m)$ is one if the class label assigned to pixel region $\mathbf{x}$ is i, else zero. and $D_v^0(\mathbf{x}) = \sum_{i=1}^{m} D_v^i(\mathbf{x})$ is the density sum over all the classes; here, $D_v^0(\mathbf{x}) = 0$ means that there are no point samples.

Visual density. Given a region Ω, we define the visual density of Ω as the proportion of nonempty space inside Ω:

$$D_v^0(\Omega) = \frac{\sum_{\mathbf{x}\in\Omega} \delta(D_v^0(\mathbf{x}) \neq 0)}{|\Omega|} \tag{1}$$

$$D_v^i(\Omega) = \frac{\sum_{\mathbf{x}\in\Omega} \delta(D_v^i(\mathbf{x}) == i)}{|\Omega|}. \tag{2}$$

where $|\Omega| = \sum_{\mathbf{x}\in\Omega} 1$ denotes the area of Ω and δ is an indicator function, which returns one, if the condition is true, and zero, otherwise.

Relative data densities. Bertini and Santucci [6] proposed the concept of *relative data densities*, which is defined on the region level. Given two regions Ω_A and Ω_B with the same area, the relative data density between them is:

$$\phi(D^0(\Omega_A), D^0(\Omega_B)) = \begin{cases} 1 & \text{if } D^0(\Omega_A) > D^0(\Omega_B) \\ 0 & \text{if } D^0(\Omega_A) = D^0(\Omega_B) \\ -1 & \text{if } D^0(\Omega_A) < D^0(\Omega_B). \end{cases} \tag{3}$$

where the comparison result is often weighted by the area density.

Relative class densities. As far as we know, the concept of relative densities has not been explored between classes, although Chen et al. [11] qualitatively evaluate multi-class sampling results by using relative densities. Following the spirit of *relative data densities*, we define *relative class densities* between two distinct classes i and j ($i \neq j$) for a common area Ω as

$$\phi(D^i(\Omega), D^j(\Omega)) = \begin{cases} 1 & \text{if } D^i(\Omega) > D^j(\Omega) \\ 0 & \text{if } D^i(\Omega) = D^j(\Omega) \\ -1 & \text{if } D^i(\Omega) < D^j(\Omega). \end{cases} \tag{4}$$

where the comparison resultis also weighted by the class density.

Preserving relative data and class densities has to be done on local regions. Typically, if $\phi(D^0(\Omega_A), D^0(\Omega_B))$ equals $\phi(D_v^0(\Omega_A), D_v^0(\Omega_B))$, we say that the output visualization preserves the relative data density between regions Ω_A and Ω_B. Similarly, if $\phi(D^i(\Omega), D^j(\Omega))$ equals $\phi(D_v^i(\Omega), D_v^j(\Omega))$, we say that the output visualization preserves the relative class density between the ith and jth class in region Ω.

Outliers. While there are no clear definitions, data points in low-density areas are often regarded as outliers [9]. For multi-class scatterplots, a point in a high-density area may also be an outlier if its class label differs from its nearby data points [39].

Since outliers are not precisely formalized, we are not going to explicitly preserve outliers, but allow users to adaptively control the sampling ratio in low-density areas. More fundamentally, preserving outliers is often in conflict with the goal of preserving relative data densities, since more point samples selected in low-density regions may distort relative data densities.

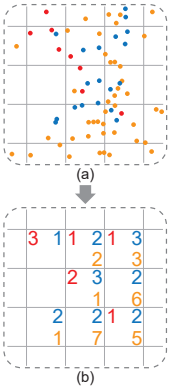


Fig. 2. (a) input scatterplot and (b) multi-class density map over 2D grid cells.

Local cells. Our sampling process in fact performs on a 2D grid of local *cells*, meaning that the pixel region $\mathbf{x}$ is a local cell region in the input scatterplot domain (see Fig. 2(a)). For a grid of $w \times h$ cells we compute the density values in the multi-class density map per local cell by counting the points inside each local cell (see Fig. 2(b)). The cell size $sz \times sz$ with unit of pixel is a user-defined parameter.

4 Sampling by Recursive Subdivision

Our goal is to design a non-uniform sampling technique that is able to faithfully preserve relative data densities and relative class densities, while at the same time trying to keep outliers. We approach the problem by formulating a two-step sampling method: first find suitable regions for placing point samples, then determine which class is to be shown in each region. Our key idea is to use a kd-tree hierarchy to encode the non-uniform density distribution so as to maximize the preservation of relative data densities, and then to take the hierarchy to guide the multi-class sampling process for preserving relative class densities. Fig. 3 gives an overview of the working pipeline of our method.

4.1 Customized KD-Tree Construction

A kd-tree is a search tree built by recursively applying axis-aligned splits to partition an n-dimensional space [21]. Such trees are mainly designed for performing efficient nearest neighbor searches [19, 22,

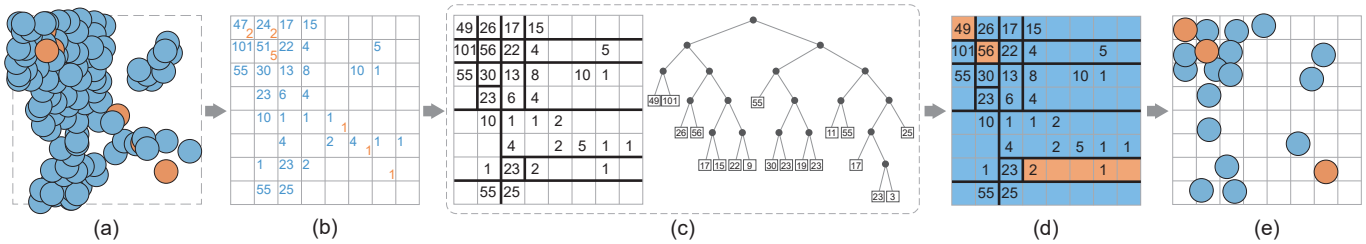


Fig. 3. Pipeline of our method: (a) the original two-class scatterplot; (b) a multi-class density map D^i is created from (a); (c) a binary kd-tree is built based on the relative data density (black lines indicate the split axes); (d) we determine the class to be shown in each leaf-node region by ensuring class visibility and locally preserving relative class densities; (e) the final point samples are randomly selected based on the results in (c).

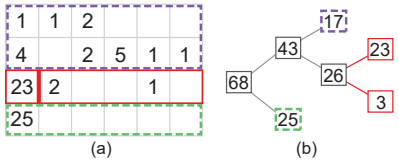


Fig. 4. An example iteration when building the binary kd-tree shown in Fig. 3(c), lower right. The split along the thick red line in (a) leads to the two new red leaf nodes boxed in (b).

23]. We employ a 2D kd-tree for choosing proper regions within a scatterplot for placing point samples that preserve relative data and class densities as well as outliers. Our customized binary kd-tree hierarchically subdivides the input density map into disjoint regions. Then we select a single point sample that is shown for each region associated to a leaf-node at the end (see Fig. 3(d & e)).

Node attributes. We denote v as a node in the kd-tree and T_v as the subtree rooted at v. When building the kd-tree, we associate the following attributes to each node (for both interior and leaf) in the tree:

- N_{leaf} is the number of leaf nodes in T_v;
- N_{cell} is the number of grid cells covered by T_v;
- N_{class} is the number of classes in the cells covered by T_v;
- N_{occupied} is the number of nonempty cells covered by T_v; and
- D_{sum} is the density sum on D^0 in the region covered by T_v.

From these attributes, we define *sampling ratio* $\alpha(v)$ for the region covered by T_v as

$$\alpha(v) = \frac{v.N_{\text{leaf}}}{v.D_{\text{sum}}}\ , \tag{5}$$

where each leaf node corresponds to a point sample in the final sampling result. Likewise, we define cell-based *visual density* $\beta(v)$:

$$\beta(v) = \frac{v.N_{\text{occupied}}}{v.N_{\text{cell}}}\ . \tag{6}$$

Subdivision criteria. To simultaneously preserve relative data densities and outliers, we introduce two criteria for guiding the recursive subdivision that builds the kd-tree. First, we explicitly restrict the difference between the sampling ratios of each two children of an inner node, so as to preserve the relative data densities between all siblings in the tree. Without loss of generality, we define the condition for the left child node v.leftchild:

$$\alpha(v.\text{leftchild}) - \alpha(v.\text{rightchild}) < \lambda\ , \tag{7}$$

where λ is a threshold parameter. If the condition is met, the v.leftchild is a candidate for a subsequent split.

A small λ requires the siblings to have almost equal sampling ratios, meaning that we cannot subdivide the corresponding child nodes, if we cannot find any split that can create two siblings with almost equal sampling ratios. Please note that we do not use the absolute difference here: $v.leftchild$ is split even when its sampling rate is λ larger than of $v.rightchild$, see Sect. 4.3. The purple box in Fig. 4(a) shows a case

Algorithm 1 Building the customized kd-tree

Input: density map D^0
Output: a kd-tree

```
function BuildTree( Node root )                          ▷ starting point
    while DivideTree( root , true ) do                   ▷ keep dividing until we cannot
        ;                                                ▷ an empty statement
    end while
end function

function DivideTree( Node v , Boolean suggestToSplit )
    if v.N_leaf > 1 then
        v_l = v.leftchild , v_r = v.rightchild
        hasSplitL = DivideTree( v_l , suggestToSplit and α(v_l) − α(v_r) < λ )
        hasSplitR = DivideTree( v_r , suggestToSplit and α(v_r) − α(v_l) < λ )
        v.N_leaf = v_l.N_leaf + v_r.N_leaf
        return hasSplitL or hasSplitR
    else
        if ((suggestToSplit or β(v) < τ) and v.N_occupied > 1) then
            [v_l, v_r] = SplitNode(v, D^0)
            v.leftchild = v_l, v.rightchild = v_r
            return true
        end if
    end if
    return false
end function
```

where no further split can be made. In such a case, we keep a single point sample to represent the data points in the entire corresponding region. We use a small λ to create different density regions with similar sampling ratioα. High density regions thus contain more samples.

The second criterion determines if a leaf node can be split to adjust the sampling ratio. It is achieved by examining each leaf node v in two steps. First, we check if there is only a single occupied cell in v (i.e., $v.N_{\text{occupied}} = 1$). If so, we cannot split v (see the green node in Fig. 4(a)). Otherwise, we check if the node meets one of the following two conditions: (i) it is suggested to be split by its parent, and (ii) its represented density $\beta(v)$ is smaller than a threshold τ.

$$\beta(v) < \tau. \tag{8}$$

The root node is split by default. If the second condition is met, a leaf has a small represented density $\beta(v)$ with more than a single nonempty cell, indicating that the related data samples are potential outliers. In this case we should try to split it, to get more leaf nodes, each helping to preserve an outlier point in the nonempty cells. In other words, the threshold τ helps preserve outliers; see Fig. 9.

KD-Tree building procedure. The process starts by creating a root node that encloses the entire density map (or scatterplot) and filling its node attributes. Then, we look for a divisible node (or region) based on the above criteria; if there is any, we divide the region by finding a proper split axis and create two new leaf nodes. If its parent does not suggest to split, we keep traversing to the leaf node to meet the sparsity condition (see Eq. (8)). After each division, we need to update N_{leaf} for all related parent nodes and then re-examine all existing interior nodes

Algorithm 2 Multi-class sampling guided by kd-tree

```
function MultiClassSampling(Tree t)
    for each leaf v in t do
        if v.N_class > 1 then
            [μ, P] = BacktrackSearch(v)
            AssignClassLabel(P, μ)
        end if
    end for each
    randomSample(t)
end function
```

from the root to determine which leaf nodes should be split. This is the major difference from traditional kd-trees, where the split is determined independently for all current leaves. Once all leaf nodes cannot be split any more, we terminate. Algorithm 1 outlines the procedure.

Splitting a node. To split a leaf node, we have to find a split axis that minimizes the density difference between the two subdivided regions. To do so, we first locate the mass center over all cells in the leaf node region weighted by their density values. Then, we consider two possible splits, i.e., a horizontal split or a vertical split across the mass center, and select the one that minimizes $|v_l.D_{\text{sum}} - v_r.D_{\text{sum}}|$, where v_l and v_r denote the two child nodes generated by a split.

Note that for each leaf node we eventually pick only one point sample to be shown in the visualization (see next subsection for details), so the sampling ratio of the purple node in Fig. 4 is 1/17 and the sampling ratio of the red node before its split is 1/26. The sampling ratio difference for the purple node and its sibling is smaller than threshold λ, i.e., $\alpha(v_l) - \alpha(v_r) < \lambda$ in Algorithm 1 becomes false, so we do not to split it. The green node also cannot be split since it has only one nonempty cell inside its region, i.e., $v.N_{\text{occupied}} > 1$ is false in Algorithm 1.

4.2 KD-Tree guided Multi-Class Sampling

After constructing the kd-tree, its leaf nodes might contain data samples from multiple classes. If we would select data samples just randomly from every leaf node, relative data densities will be preserved, but we potentially would miss points of rare classes. Fig. 5(b) shows an example, where all orange nodes shown in subfigure (a) are lost. On the other hand, it is possible to give high priority to samples of the rare classes by first assigning class labels to leave nodes with rare classes. This, however, might distort the relative class densities. As an example, the three orange points in the top-left corner of Fig. 5(c) are shown, but the dense points of the blue class are lost.

To address these issues, we propose to use a larger part of the kd-tree structure to guide the multi-class sampling process rather than individually checking only the leaf nodes. To ensure that the data samples of all existing classes are shown, while minimizing the distortion of relative class densities, we propose a three-step sampling strategy that first backtracks each leaf node to an ancestor node that has enough point samples, and then assigns class labels to each leaf node of the local sub-tree. Finally, we randomly sample a single data sample in each leaf with the assigned class label. Fig. 6 illustrates this strategy with two examples; in the following, we discuss the details of the algorithm.

Backtracking search. For each leaf node v that has data samples from multiple classes, we backtrack to an ancestor node in order to maximize the preservation of relative class densities.

Suppose the ancestor node μ has k leaves that are associated with samples from c ($c \leq m$) classes, where the number of samples in each class is nds_i. If $k < c$, we continue and visit its parent node; otherwise we measure the faithfulness of this node in preserving relative class densities. Specifically, we first collect one point sample from each of the c classes, and then randomly select class labels for the remaining $k - c$ leaves using roulette wheel selection [32]. Hence, we obtain a class allocation array $\mathbf{P} = \{p_1, \cdots, p_m\}$, where p_i is the number of point samples of the i-th class and $\sum_i^m p_i = k$. After sorting the array $\{nds_1, \cdots, nds_c\}$ in descending order, we compute the consistency between the data and visual density for all pairs of classes by

$$\omega(\mu) = \frac{\sum_i^m \sum_{j<i} \rho_{i,j} \delta(\phi(D^i(\mu), D^j(\mu)), \phi(p_i, p_j))}{\sum_i^m \sum_{j<i} \rho_{i,j}}, \qquad (9)$$

where $\rho_{i,j} = \frac{nds_i}{nds_j}$ is the data density ratio between classes i and j, and δ returns one if the relative class density is preserved, else zero. By testing different interior nodes, we choose the one with the largest ω.

Fig. 7 illustrates this procedure, where the consistency values of nodes A and B are both one, but we prefer node B, since the deeper the interior node is, the easier it is to maintain the relative class densities, because the effect of keeping rare classes in the visual representation will diminish. In addition, backtracking applied an large parts of the tree can be quite slow. Thus, we set the maximum search depth to four by default. Fig. 8 shows the influence of different values.

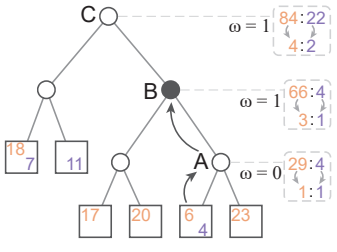


Fig. 7. Backtracking procedure.

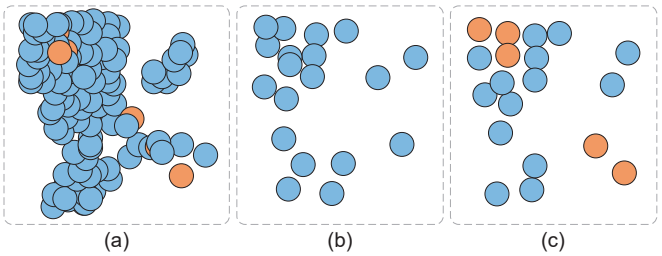


Fig. 5. Two straightforward strategies for kd-tree based multi-class sampling. (a) Input scatterplot (from Fig. 3(a)); (b) random sampling produces a point set where all orange points are missed; (c) giving high priority to rare class helps preserve them, but changes relative class densities.

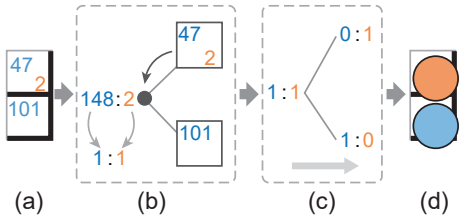
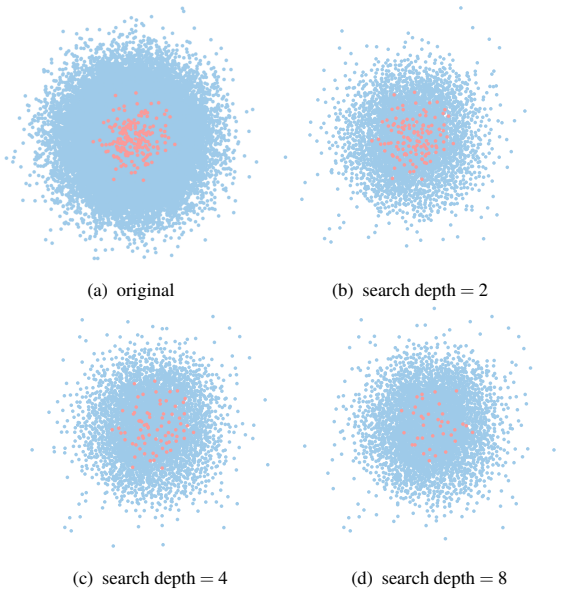


Fig. 6. Illustration of our three-step multi-class sampling. A sub-region was extracted from the tree in Fig. 3(c); (b) Backtracking from the leaf to an interior node in the sub-tree; (c) assigning a class label to each leaf so that all classes are covered; and (d) final sampling result.

Class label assignment. Starting from the selected interior node μ and the class allocation array P, we recursively compute such arrays for the children until all leaf nodes have an assigned class label. At each step, we first compute two class allocation arrays P_l and P_r for the two children of μ. In order to preserve rare classes, we select the node with the larger N_{class} for the class label assignment at the first place. Once the array P_i for the selected node is created, the class allocation array for the other node can be obtained by computing the difference between P and P_i. During the assignment, we first allocate one leaf node for each of the classes that do not appear in the sibling node and then randomly select class labels for the remaining leaf nodes.

Random sampling. After assigning a class label l to each leaf node, we randomly sample one point from the data samples with the label l associated with this leaf. Doing so, the final sampling result is obtained as shown in Fig. 3(e).

Fig. 6 shows the process for a sub-tree of Fig. 3(c). Since we have much more blue than orange samples, a random assignment due to relative densities would assign blue to both leaves, while our algorithm first backtracks to the parent node and then guarantees that at least one



(a) original (b) search depth = 2 (c) search depth = 4 (d) search depth = 8

Fig. 8. Effect of backtracking depth on a synthetic dataset with two Gaussian-distributed classes (40k blue points and 200 red points). The red class is rendered purposely upon the blue one to show its structure clearly. A small search depth value is beneficial to keep rare classes but also amplifies them, a larger value converges to random sampling.

leaf will be orange. In doing so, each leaf has been assigned one point sample, one from the orange class and the other from the blue.

4.3 Time Complexity & Parameter Analysis

Time complexity. As shown in Algorithm 1, to determine every new split, we have to traverse the tree from the root to the leaves, involving $\log(n)$ computations of the mass centers with a time complexity of $O(n)$ (n is the number of grid cells). Thus, the time complexity of each division is $O(n \log n)$. Given a scatterplot with n non-empty cells, the overall time complexity is $O(n \log^2 n)$. Note that n is independent of the number of data samples, due to our grid-based structure.

Cell size. The cell size sz determines the number of grid cells, where too few grid cells will make it hard to perceive density differences in dense areas of the plot, while too many grid cells will lead to significant overdraw. A good heuristics for the cell size is to have it in the order of the radius of the drawn circles for the points (in pixels). Figs. 9(a-c) shows the effect of different grid sizes while all other parameters of the algorithm are constant. In our experiment, we set the size as 6×6 for the *Person Activity* data set in Fig. 1 and Figs. 9(d-i).

Parameters λ and τ. Parameter λ influences how many interior nodes will be split during the recursive subdivision; see examples in Figs. 9(d-f). For a high λ, e.g., $\lambda = 1$, both children of the interior nodes are likely to be split and the subdivision will reach the bins of grid, so all outliers will be visible, but relative densities will be disturbed in dense areas. If we set $\lambda = 0$, only one of the children will be split every time, the algorithm will slow down and outliers will become less visible.

In contrast, τ mainly influences the results in sparse areas. As shown in Figs. 9(e,g,h), a large τ reveals more outliers, but changes relative class densities between dense and sparse regions. Hence, an extremely large λ and τ would result in strong overdraw and strong disturbances of relative data densities; see the example in Fig. 9(i). Depending on the application, we have to find good parameter values, e.g., we set both λ and τ to 0.02 for the *Person Activity* data set in Fig. 1.

5 Evaluation

This section presents a quantitative comparison with the state-of-the-art sampling methods and three case studies on both synthetic and real datasets, and an extension for scatterplot matrix on a PC with an Intel i5-7400 3GHz CPU and 24GB RAM. All density maps are 1600×900.

5.1 Quantitative Evaluation

We implemented all the above-mentioned sampling methods: random sampling, non-uniform sampling [4], multi-class blue noise sampling [11] (blue noise for short), and our method, in C++. Unless clearly specified, all data samples in our scatterplot results are rendered in a random order. Since random sampling and non-uniform sampling both do not exploit class information, we take our multi-class scatterplots as single-class ones as their input and randomly sample a class sample from each point. For a fair comparison, we adapt them from the original pixel level sampling into our local cell level. Blue noise sampling heavily depends on the density field, so we adopted Silverman's rule of thumb [41] to determine the bandwidth, but this might not be exactly the same as the authors' version [11]. By default, we set the parameters of our method to the following values: $sz = 6, \lambda = 0.02, \tau = 0.02$, and adjusted the parameters of the other sampling methods to obtain a similar number of point samples as with our method.

Datasets. For a comprehensive evaluation, we collected 37 labeled datasets that are substantially different from Kaggle [24] and the UCI data repository [31] in terms of the number of data samples (ranging from 4K to 1,6M) and number of classes (2 to 18). Among them, 10 synthetic datasets were manually created with random Gaussian classes, and 27 real datasets were collected from the UCI data repository [31] and Kaggle [24].

Metrics. Since our approach attempts to preserve relative data density, relative class density, and outliers, we employ the following four metrics to quantitatively evaluate the quality of the sampled results:

- *Perceived Data Densities ratio* (**PDDr**) is adopted from Bertini and Santucci [4]. Given two regions Ω_A and Ω_B of the same area, it measures how much relative data densities are preserved by visual densities

$$\text{PDDr} = \frac{\sum_i \sum_{j<i} \chi_{ij} \delta(\phi(D^0(\Omega_i), D(\Omega_j)), \phi(D_v^0(\Omega_i), D_v^0(\Omega_j)))}{\sum_i \sum_{j<i} \chi_{ij}}, \tag{10}$$

  where $\chi_{ij} = D^0(\Omega_i) + D^0(\Omega_j)$, and δ returns 1 if the relative data density between Ω_A and Ω_B is preserved, else 0. The range of *PDDr* is $[0,1]$, where a large PDDr ~ 1 means better relative data density preservation. PDDr looks similar to $\omega(\mu)$ in Eq. (9), but $\omega(\mu)$ is defined on relative class densities in the region μ.

- *Perceived Class Densities ratio* (**PCDr**) aims to evaluate how a sampling method preserves relative class densities. Here, we formulate the metric PCDr based on the weighted Spearman's ranking correlation coefficient γ_s, which has been widely used for measuring the rank correlations [20]. Given a region Ω_i, we compute two ranks x_k and y_k of each kth class in terms of data density $D(\Omega_i)$ and visual density $D_v(\Omega_i)$ and then compute the Spearman correlation coefficient by:

$$\gamma_s(\Omega_i) = 1 - \frac{6 \sum_k (x_k - y_k)^2}{m(m^2-1)},$$

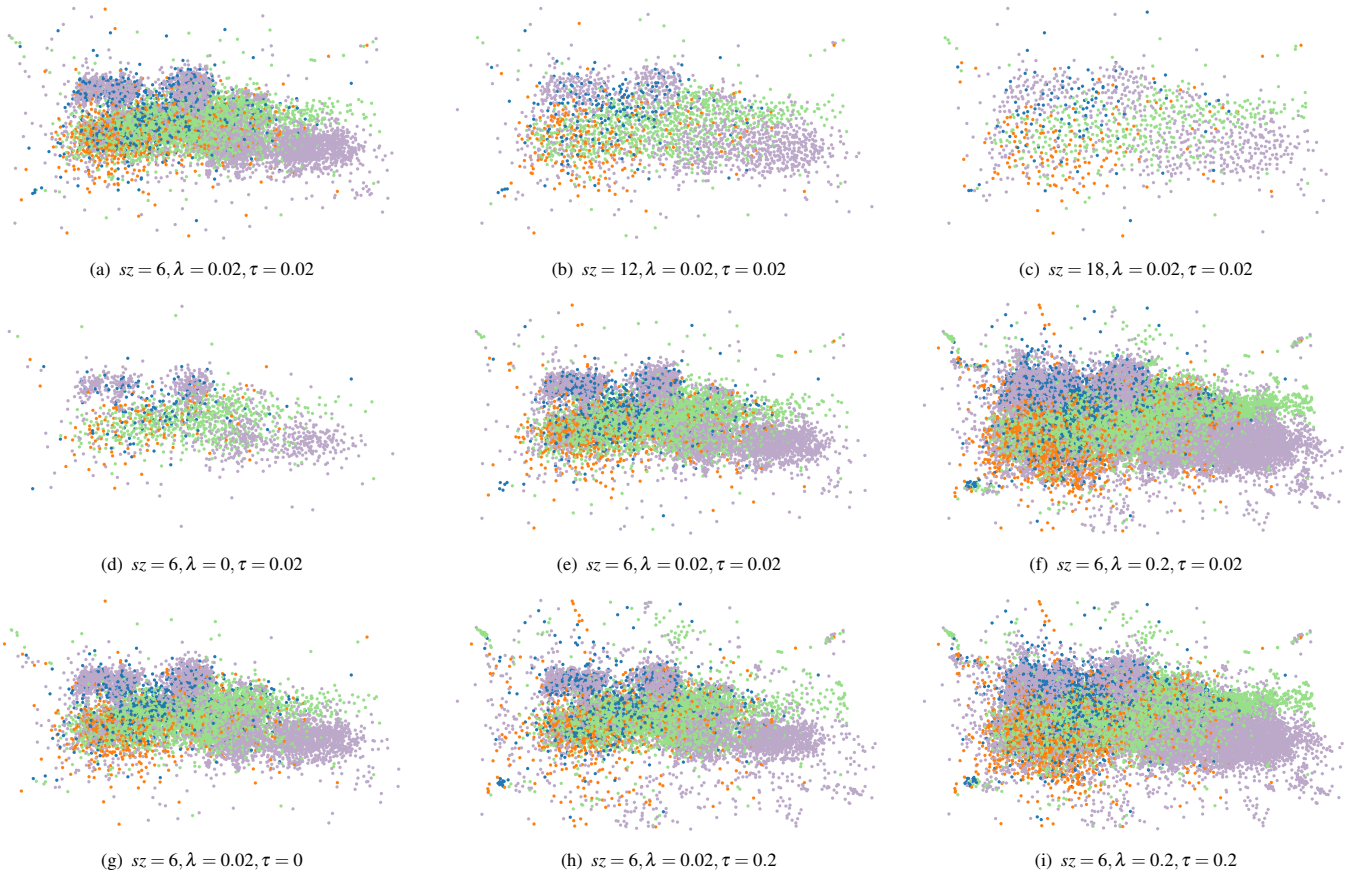
  where γ_s is a value between 0 and 1 and m is the number of classes. By weighting γ_s with the density value $D^0(\Omega_i)$, we compute the change in the relative class density over the whole scatterplot by

$$\text{PCDr} = \frac{\sum_i D^0(\Omega_i) \gamma_s(\Omega_i)}{\sum_i D^0(\Omega_i)}. \tag{11}$$

- *Erased Sample Regions ratio* (**ESRr**) is adopted from Bertini and Santucci [4] that measures how many outliers are lost in the low-density areas. Specifically, it computes the portion of empty regions caused by sampling:

$$\text{ESRr} = \frac{\sum_i \delta(D^0(\Omega_i), 0)}{N}, \tag{12}$$

  where N is the number of sample regions. Hence, a small ESRr indicates a better outlier preservation.



(a) $sz = 6, \lambda = 0.02, \tau = 0.02$ (b) $sz = 12, \lambda = 0.02, \tau = 0.02$ (c) $sz = 18, \lambda = 0.02, \tau = 0.02$

(d) $sz = 6, \lambda = 0, \tau = 0.02$ (e) $sz = 6, \lambda = 0.02, \tau = 0.02$ (f) $sz = 6, \lambda = 0.2, \tau = 0.02$

(g) $sz = 6, \lambda = 0.02, \tau = 0$ (h) $sz = 6, \lambda = 0.02, \tau = 0.2$ (i) $sz = 6, \lambda = 0.2, \tau = 0.2$

Fig. 9. Parameter analysis on the *Person Activity* data set. (a,b,c) Grid size influences the number of point samples. From left to right, the results have 5969, 2273, and 1217 points, respectively. (d,e,f) For a large λ, many outliers become visible, but overdraw happens in dense areas, while a small λ reduces overdraw but miss a few outliers. (g,e,h) A large τ shows too many outliers and regions of medium density are suppressed, while a small τ is more balanced but outliers are reduced. (i) When λ and τ both are large, the overdraw issue becomes severe while showing many outliers.

- *Erased Class Sample ratio* (**ECSr**) is designed for measuring how many rare classes are lost after sampling (samples of rare classes are also regarded as outliers in multi-class scatterplots). For each region Ω_i, we count the number of classes before and after sampling, obtain the difference $\rho(\Omega_i)$ and finally count all lost classes due to sampling:

$$\text{ECSr} = \frac{\sum_i D^0(\Omega_i)|\rho(\Omega_i)|}{\sum_i D^0(\Omega_i)}. \tag{13}$$

Results. We measure the results of all sampling methods with a region size of 40 $\times$ 40. Screen shots of the results generated by all four sampling methods on each dataset and the according scores for each metric can be found in the supplemental material. Fig. 10 gives an overview of the scores of all metrics for all datasets. Our method seems to outperform the others in terms of PCDr and ECSr, indicating our method is the best in preserving relative class densities and rare classes. On the other hand, it is ranked as the second in terms of the other two metrics and performs very similarly to the best methods. This result is expected and confirms that our method makes a good balance between preserving relative data densities and class densities.

Since random sampling uniformly samples a scatterplot, it certainly performs the best in terms of PDDr, but it is the worst w.r.t. outlier preservation, see Fig. 10(c). In contrast, non-uniform random sampling is the best in maintaining outliers, because it explicitly preserves all low density regions. However, such preservation might loose relative data densities, see the PDDr scores in Fig. 10(a). Both methods are not intentionally designed for multi-class data, but their scores in PCDr and ECSr are better than blue noise sampling, which is ranked last. We think the reason is that the artifacts introduced by blue noise sampling method severely hurts the proper representation of relative data and class densities (seen Fig. 1(d)). Instead, our multi-class sampling method more faithfully represents all the wanted characteristics, see Section 5.2.

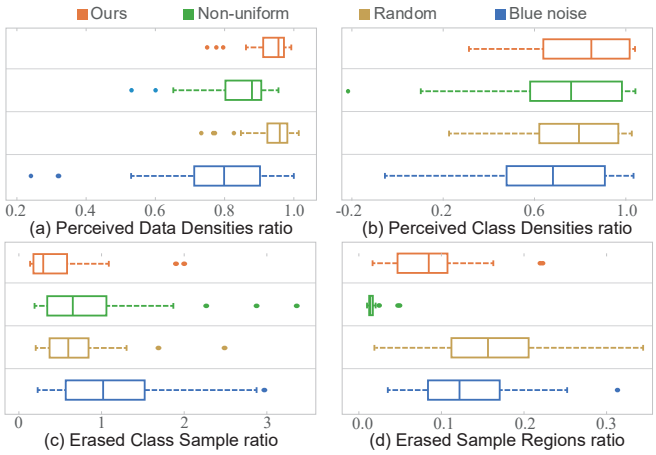


Fig. 10. Boxplots summarizing the scores of four measures for four sampling methods over all tested datasets: (a) PDDr and (b) PCDr, where a higher score indicates better sampling for both measures; (c) ESRr and (d) ECSr, where a lower score indicates better sampling.

Runtime. Fig. 11(a) shows the log-scale runtime of the four methods on all datasets. Non-uniform sampling seems to perform the best, followed up by random sampling and our method, while the blue noise sampling is more than 1000 times slower than the others on average. After carefully looking at the runtime for each dataset, we found that

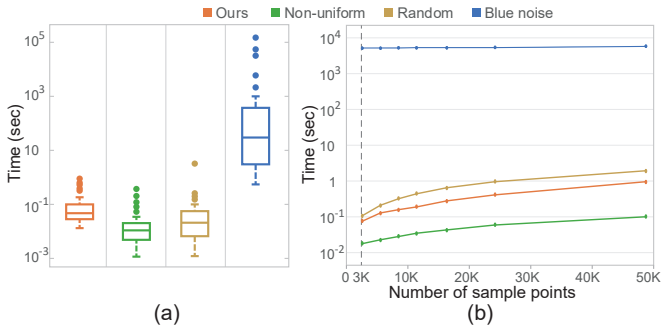


Fig. 11. (a) Runtimes of the four sampling methods; (b) relationship between runtime and sampled points (logarithmic scale).

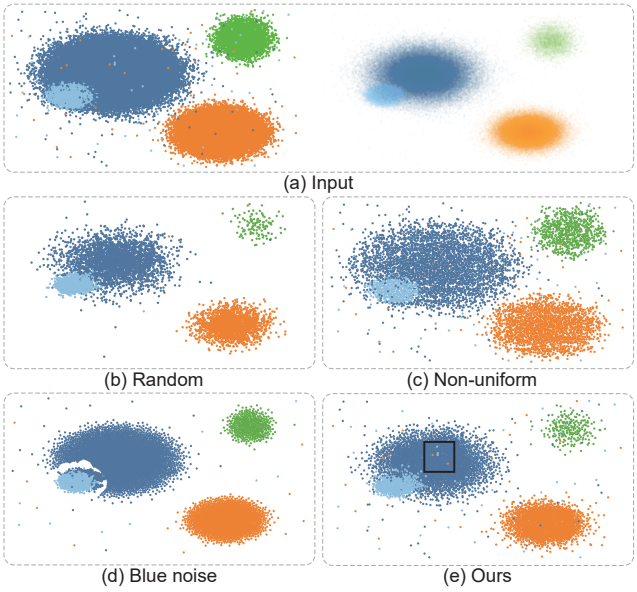


Fig. 12. Sampled results of the synthetic dataset with 250K points. (a) original scatterplot (left) and four classes sampled individually (right), the transparency of each point is proportional to its density value; (b,c,d,e) results for 6K points generated by random sampling, non-uniform sampling, blue noise sampling, and our method.

the blue noise method requires around 20 mins for 100K data samples and 27 hours for 1,560K data samples, because it heavily depends on the number of data samples, cf. [45]. In contrast, the other three methods handle all data almost in less than 1 second.

To understand how the number of sampled points influences the runtime, we applied the four methods to a synthesized dataset (250K points) with varying amount of sampled points and run the algorithm multiple times for each setting. Fig. 11(b) shows the results and indicates that the running time of the blue noise sampling is not related to the number of sampled points, while the runtimes of the other three methods gradually increase as the number of sampled points increases but still are below a 1 second even for 50K points. This means that our method is able to efficiently work with large scale datasets.

5.2 Case Studies

We conducted case studies with one synthetic dataset and two real datasets from daily life and computer science.

5.2.1 Synthetic data

To ensure that our method is able to preserve outliers and rare classes, we synthesized a dataset with 250K data points and four separated Gaussian clusters. A small number of points coming from the other three classes were mixed among the point distribution of each class. In order to represent rare classes in each region, we sort all classes in terms of numbers of points in descending order and render the classes

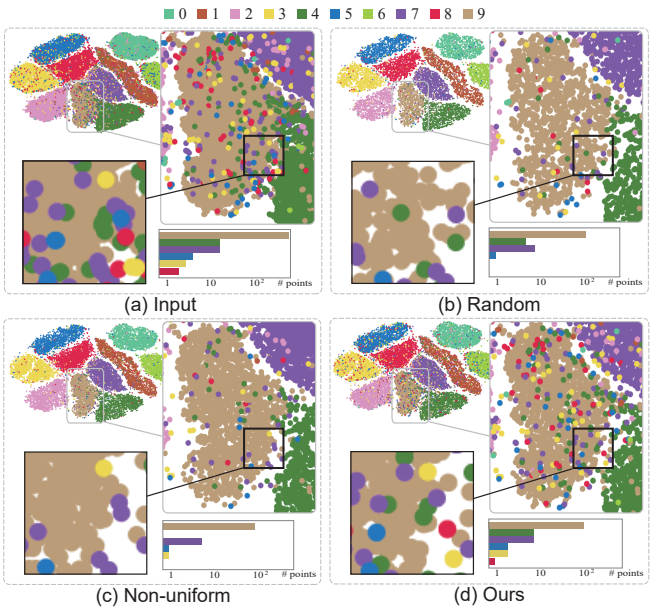


Fig. 13. Sampled result of the MNIST dataset: (a) original scatterplot with 70K data points; (b,c,d) sampled results generated by random sampling, non-uniform sampling and our method.

in this order. Fig. 12(a) shows the input scatterplot (left) and each of the individual classes (right), where the dark blue cluster is denser than the green cluster but is similar to the other two clusters.

Figs. 12(b,c,d,e) show the sampled results of the four methods with around 6K points. Density differences between the dark blue cluster and the green cluster are well preserved in Figs. 12(b,e), but are lost in Figs. 12(c,d). The non-uniform sampling almost keeps all outliers as shown in Fig. 12(c), but causes a deficiency in preserving relative data densities. Blue noise sampling shows the density of the green cluster similar to the other clusters and introduces a gap between the light blue and dark blue clusters, which does not appear in the input and the other results. Because of such artificial patterns, the PCDr score in Fig. 12(d) is much lower than for the other methods. Compared to Fig. 12(b), our method almost preserves all major outliers in Fig. 12(e), while non-uniform sampling in Fig. 12(c) seems to show even too much. We can spot a few outliers from rare classes in our result, e.g., the ones shown with a black box in Fig. 12(e), which are not presented by the other methods. In this sense, our method seems to be best in preserving rare classes while balancing the other characteristics well.

5.2.2 Real Data

Since the blue noise method is quite slow and furthermore seems to create some unwanted patterns as shown in Fig. 1(d) and Fig. 12(d), we do not include it for this study.

MNIST. Exploring dimensionality reduction (DR) results with scatterplots is a common practice in data analysis [40]. For a large dataset, working with the sampled version with well-preserved data characteristics can greatly improve the efficiency. Here, we explore the DR result for the MNIST dataset [28], which consists of 70K samples and 10 classes with handwritten digits from 0 to 9. Like Fig. 12, we render the rare classes on top. Fig. 13(a) shows the original scatterplot produced by t-SNE, where the major structure of each class is clearly revealed but is contaminated by a few points from other classes. Since such points heavily influence the classification accuracy, it is important to preserve them in the sampled result. Figs. 13(b,c,d) show the results generated by random sampling, non-uniform sampling and our method. In terms of ECSr, our method (0.16) performs much better than the other methods (1.12).

To further study the properties of our method, we selected an example region (denoted by the black frame in Fig. 13) and counted the number of points of each class in this area (shown as bar charts in the

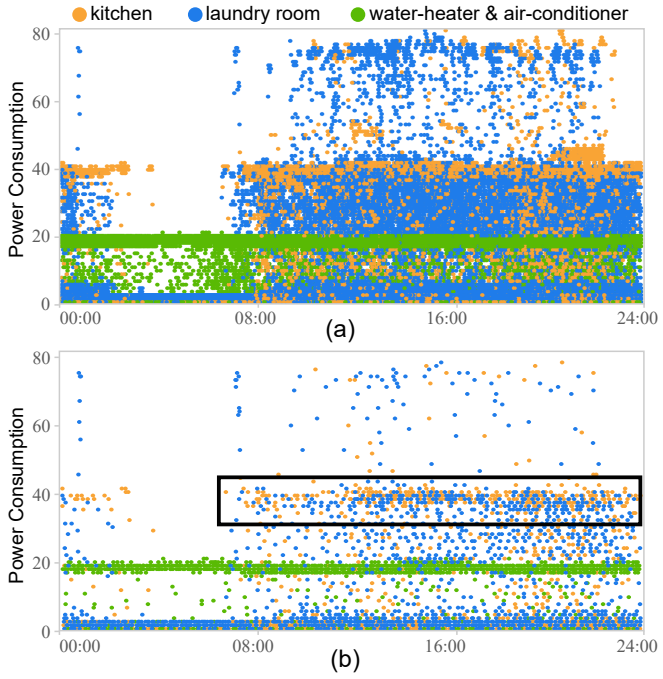


Fig. 14. Exploration of the power consumption dataset with 1,570K data points: (a) the original scatterplot; (b) the sampled result generated by our method.

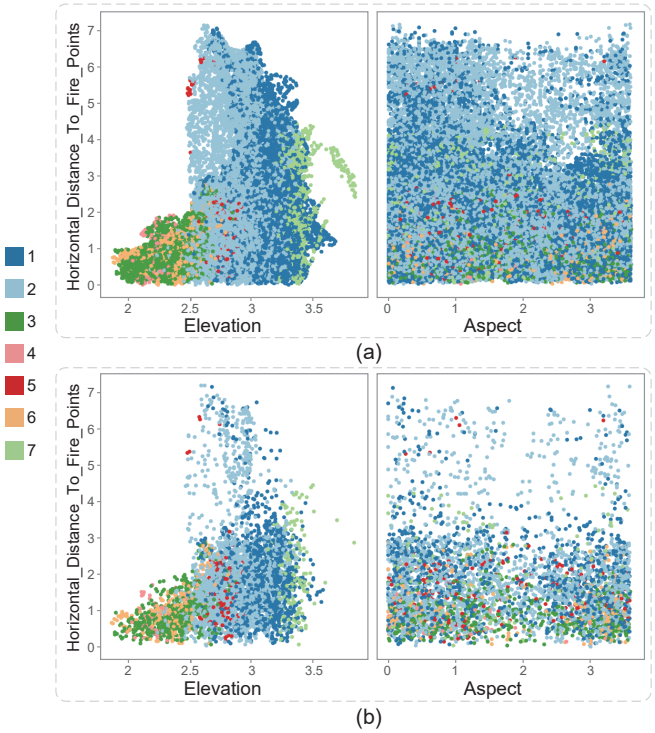


Fig. 15. Sampling of a multi-class scatterplot matrix. (a) Two input scatterplots with 50K data samples defined by the selected three variables; (b) the corresponding two sampled scatterplots with 4K samples.

right bottom of each subfigure). The bar charts indicate that our method keeps most points of rare classes while preserving relative class ratios among them.

Electric Power Consumption. To study the effectiveness of our method for handling large data sets, we applied our method to the *electric power consumption* [8] dataset, which contains around 1,570K measurements of electric power consumption gathered from a house in each minute. Those measurements are classified into 3 classes: power consumption in kitchen, laundry room, and on an electric water-heater as well as an air-conditioner.

A scatterplot with the variables of power consumption (y axis) and time (x axis), cf. Fig. 14(a), shows that the laundry room is heavily used from 8:00 to 24:00, and the points are equally distributed at all levels of power consumption. For the rest of the day a constant low power consumption seems to appear, which is counter intuitive to our daily life. By applying our sampling method to this scatterplot (Fig. 14(b)) we receive a sampled scatterplots with 3K points, which reveals three interesting findings that can be hardly spotted in Fig. 14(a). First, the blue points in the black box denote the laundry room usage is mostly from 12:00 (noon) to 20:00 with a power consumption of 40Wh. This indicates that the householder mostly does laundry in the afternoon and evening, meanwhile the power consumption of the washing machine is most likely to be 40Wh. Second, the green points are mainly at 20Wh all day long, and the rest of the green points are sparsely distributed under 20Wh. This seems to indicate that the air-conditioner might be functioning all day and its power consumption is about 20Wh. Both patterns are not clearly visible in Fig. 14(a). In addition, we see that the constant low consumption under 5Wh in the laundry room is not just from 0:00 to 8:00, but lasts for the whole day. After further investigating the input data, all patterns are confirmed, the constant consumption under 5Wh comes from the refrigerator.

5.3 Extension for Scatterplot Matrix

Finally, we extend our method for sampling multi-class scatterplot matrix. Taking the multi-dimensional data as an input, we define the cells in multi-dimensional space and then construct a multi-dimensional kd-tree for multi-class sampling. Note that the splitting axis is selected from all related axes, not only from the x and y axes anymore. Fig. 15 shows an example with two scatterplots of three selected variables. Our sampled results show the spatially-varying densities of the two blue classes, especially the top part, while preserving outliers of the red class in both subfigures. Results of the full scatterplot matrix can be found in the supplemental material.

6 Conclusion

We presented a recursive subdivision technique for sampling multi-class scatterplots, that allows to preserve relative data densities while faithfully maintaining relative class densities and showing major outliers. It is achieved by first subdividing the multi-class density field into a customized binary kd-tree and then using this kd-tree to perform multi-class sampling. We conducted a quantitative evaluation to compare our method with the state-of-the-art techniques, and present three case studies and an extension for scatterplot matrix, which helps exploring multi-dimensional data.

There are still some limitations in our technique. First, our kd-tree-guided sampling does not take into account the spatial neighborhood information which might cause gaps between regions. Second, we only tested a few examples of scatterplot matrices and would like to investigate applications in sampling of multi-class parallel coordinates and star coordinates in the future. Last, we would like to conduct a large user study to confirm that our method is able to sufficiently support users in completing analysis tasks such as identifying outliers and characterizing distributions.

Acknowledgments

This work is supported by the grants of the National Key Research & Development Plan of China (2016YFB1001404), NSFC (61772315, 61861136012), NSFC-Guangdong Joint Fund (U1501255), Science Challenge Project (TZ2016002), the Leading Talents of Guangdong Program (00201509), the DFG Center of Excellence 2117 "Centre for the advanced Study of Collective Behaviour" (ID: 422037984), the DFG Project 493/19 "Perception-based Information Visualization," and Shenzhen Science and Technology Program (No. JCYJ20170413162617606).

REFERENCES

[1] C. Ahlberg. Spotfire: an information exploration environment. *ACM SIGMOD Record*, 25(4):25–29, 1996. doi: 10.1145/245882.245893

[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. doi: 10.1111/cgf.13446

[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] C. Blake and C. J. Merz. UCI repository of machine learning databases. `https://archive.ics.uci.edu/ml/datasets.html`, 1998.

[9] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: identifying density-based local outliers. In *Proc. ACM SIGMOD International Conference on Management of Data*, vol. 29, pp. 93–104. ACM, 2000. doi: 10.1145/342009.335388

[10] 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

[11] 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

[12] 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

[13] T. N. Dang, L. Wilkinson, and A. Anand. Stacking graphic elements to avoid over-plotting. *IEEE Trans. Vis. & Comp. Graphics*, 16(6):1044–1052, 2010. doi: 10.1109/TVCG.2010.197

[14] 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

[15] 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

[16] 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

[17] R. Fattal. Blue-noise point sampling using kernel density model. *ACM Trans. Graph. (SIGGRAPH)*, 30(4):48, 2011. doi: 10.1145/2010324.1964943

[18] 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

[19] Y. Frishman and A. Tal. Multi-level graph layout on the gpu. *IEEE Trans. Vis. & Comp. Graphics*, 13(6):1310–1319, 2007. doi: 10.1109/TVCG.2007.70580

[20] E. Frøkjær, M. Hertzum, and K. Hornbæk. Measuring usability: are effectiveness, efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pp. 345–352. ACM, 2000. doi: 10.1145/332040.332455

[21] H. Fuchs, Z. M. Kedem, and B. F. Naylor. On visible surface generation by a priori tree structures. In *Proc. SIGGRAPH*, vol. 14, pp. 124–133. ACM, 1980. doi: 10.1145/800250.807481

[22] T. Itoh, Y. Yamaguchi, Y. Ikehata, and Y. Kajinaga. Hierarchical data visualization using a fast rectangle-packing algorithm. *IEEE Trans. Vis. & Comp. Graphics*, 10(3):302–313, 2004. doi: 10.1109/TVCG.2004.1272729

[23] J. Jo, J. Seo, and J.-D. Fekete. Panene: A progressive algorithm for indexing and querying approximate k-nearest neighbors. *IEEE Trans. Vis. & Comp. Graphics*, 2018. doi: 10.1109/TVCG.2018.2869149

[24] Kaggle Inc. Kaggle. `https://www.kaggle.com/`.

[25] 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

[26] 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

[27] M. Krzywinski and B. Wong. Points of view: plotting symbols, 2013. doi: 10.1038/nmeth.2490

[28] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791

[29] J. A. Lee and M. Verleysen. *Nonlinear dimensionality reduction*. 2007. doi: 10.1007/978-0-387-39351-3

[30] 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

[31] M. Lichman. UCI machine learning repository, 2013.

[32] A. Lipowski and D. Lipowska. Roulette-wheel selection via stochastic acceptance. *Physica A: Statistical Mechanics and its Applications*, 391(6):2193–2196, 2012. doi: 10.1016/j.physa.2011.12.004

[33] 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

[34] 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

[35] 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

[36] 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

[37] T. Munzner. *Visualization analysis and design*. AK Peters/CRC Press, 2014. doi: 10.1201/b17511

[38] 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

[39] A. Sarikaya and M. Gleicher. Scatterplots: Tasks, data, and designs. *IEEE Trans. Vis. & Comp. Graphics*, 24(1):402–412, 2018. doi: 10.1109/tvcg.2017.2744184

[40] 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

[41] B. W. Silverman. *Density Estimation for Statistics and Data Analysis*. Routledge, 1986.

[42] E. R. Tufte. *The visual display of quantitative information*, vol. 2. Graphics press Cheshire, CT, 2001.

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

[44] 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

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

[46] L. Wilkinson. *The grammar of graphics*. Springer, New York, NY, 2006. doi: 10.1007/0-387-28695-0

[47] 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