

De-cluttering Scatterplots with Integral Images

Hennes Rave, Vladimir Molchanov, and Lars Linsen

Abstract—Scatterplots provide a visual representation of bivariate data (or 2D embeddings of multivariate data) that allows for effective analyses of data dependencies, clusters, trends, and outliers. Unfortunately, classical scatterplots suffer from scalability issues, since growing data sizes eventually lead to overplotting and visual clutter on a screen with a fixed resolution, which hinders the data analysis process. We propose an algorithm that compensates for irregular sample distributions by a smooth transformation of the scatterplot's visual domain. Our algorithm evaluates the scatterplot's density distribution to compute a regularization mapping based on integral images of the rasterized density function. The mapping preserves the samples' neighborhood relations. Few regularization iterations suffice to achieve a nearly uniform sample distribution that efficiently uses the available screen space. We further propose approaches to visually convey the transformation that was applied to the scatterplot and compare them in a user study. We present a novel parallel algorithm for fast GPU-based integral-image computation, which allows for integrating our de-cluttering approach into interactive visual data analysis systems.

Index Terms—Scatterplot, integral image, regularization.

1 INTRODUCTION

Visual representation of multidimensional data has been and remains a challenging task, which becomes more demanding as the dimensionality and size of datasets steadily grow. *Scatterplots* are an effective and thus widely used method for visualizing multidimensional data in a 2D domain by relating pairs of data dimensions. Scatterplots reveal the structure of the data including outliers, clusters, patterns, and tendencies.

When rendering scatterplots within a visual domain on a screen with a fixed resolution, growing numbers of data samples eventually lead to occlusion and *overplotting*, negatively affecting user perception and hindering visual data analysis. In particular, it becomes difficult to visually estimate the number of samples and their density in cluttered regions. Moreover, access to individual data samples is restricted, which impedes user exploration, e.g., of image data collections [1].

Several approaches were proposed to alleviate the overplotting issue. A common strategy is to adjust the transparency of the displayed samples to improve the visibility of the local density of the points' distribution [2]. Alternatively, the number of rendered elements can be reduced by applying a down-sampling strategy to the dataset [3]–[5]. Such approaches change the appearance of the presentation.

All authors are with the University of Münster, Germany.
E-mail: {hennes.rave|molchanov|linsen}@uni-muenster.de.
Manuscript received xxxx; revised xxxx.

Another broad class of methods uses *spatial distortions* of the representation to reduce overplotting. These techniques may remap samples to pixels or distort the visualization domain for more efficient use of the screen space.

We propose a numerical method for a smooth iterative deformation of the scatterplot domain aiming at optimizing the available plotting space usage. In each iteration, the per-pixel deformation is constructed based on a set of density integrals, so-called *integral images* (*InIm*s). Here, a crucial difference of the proposed method from existing deformation approaches is that the integrals are not restricted to any local neighborhood but rather characterize global density distribution. Therefore, the displacement of each sample depends on the global data distribution encoded in *InIm*s rather than the samples' distribution in its neighborhood. As a result, no expensive collision detection is needed and large datasets (i.e., datasets with the number of points close to the screen resolution or even beyond) can be regularized at highly interactive rates. The transformed scatterplot has a nearly regular sample distribution, significantly mitigating the overplotting issues, and making large amounts of samples visible and manageable.

Our proposed deformation preserves essential properties of the original scatterplot such as *neighborhood relations* of the displayed samples, including their local ordering, which does not automatically hold in parameter-sensitive smoothing-based regularizations. In contrast to sampling approaches, the proposed algorithm is *deterministic* and *preserves all data samples* in the visualization domain. We show that, in comparison to opacity-based approaches, the alternative regularized view allows the user to better perceive the density and quantity of data samples, more easily analyze class-cluster relations, and more easily retrieve further information about individual samples.

Our regularized scatterplots are meant to complement the original scatterplots. While the original scatterplots convey the global data distribution including clusters and outliers, our deformed versions give access to detailed structures that were occluded in the original scatterplot. We support continuous transitions between the two views, following the *reconfigure* interaction principle [6]. Additionally, we propose several approaches for *conveying initial distribution* of samples of the original scatterplot in the deformed scatterplots. We present grid lines, background texture, and contour lines and compare them with each other.

The primary target applications of our approach are within interactive visual analyses, where the user controls the desired level of deformation. Other interactions benefit

from our regularization of the samples' distribution. For example, less zooming would be required after regularization, and individual samples would be more accessible, e.g., for selections via brushing, clicking, or hovering. The *interactivity* constraints of the visualization system demand high computational efficiency of our proposed algorithm and good computational scalability. We satisfy these requirements by proposing a novel algorithm that enables fast GPU implementation of all necessary computations.

Our main contributions can be summarized as follows:

- We provide a novel deterministic technique for de-cluttering scatterplots using integral images (InIm_s), which substantially improves stability, convergence, and efficiency of a prior algorithm [7]. We compare our approach to the state of the art for visual clutter reduction in scatterplots. Our GPU implementation is made publicly available.
- We apply the algorithm for a smooth transformation of the visual scatterplot domain, where the level of deformation can be adjusted by the user.
- We propose different visual encodings to convey the amount of local deformations, information about the cluster structure, and the samples' density in the deformed plots. We evaluate their effectiveness in a controlled user study.

We provide respective background information and introduce required terms and notations in Section 3. In Section 4, we present the basic deformation formula as presented by Molchanov and Linsen [7], discuss its limitations, and provide the key steps of our deformation algorithm including GPU implementation details. Section 5 is dedicated to the proposed visual encodings, which serve to compensate for the local samples' density information typically lost after deformation. Results of our numerical tests and details on the conducted user study are provided in Section 6.

2 RELATED WORK

Scatterplots are arguably the most commonly used visualization method for data sets with more than one numerical dimension and have a long-standing history [8]. Traditional scatterplots relate two data dimensions by drawing data samples as points in a 2D Cartesian coordinate system. Frequently, the original data are multidimensional and some dimensionality reduction technique precedes the visualization step. The two scatterplot dimensions can, thus, be selected from the given set of data attributes or may represent a linear combination of them like in principal component analysis [9] or, more generally, in star-coordinates projections [10], [11]. Moreover, they could also be a 2D embedding of the multidimensional data using non-linear dimensionality reduction methods such as multidimensional scaling (MDS) [12], [13], t-distributed stochastic neighbor embedding (t-SNE) [14] or uniform manifold approximation and projection (UMAP) [15]. An extensive user study on the combination of various dimensionality reduction techniques with scatterplot representations for data exploration tasks was performed by Sedlmair et al. [16].

While conventional scatterplots render data samples as points, different approaches propose to incorporate addi-

tional information using, e.g., *glyphs* or summary visualizations. Chan et al. [17] enhanced scatterplots with sensitivity coefficients representing local correlations of data. Janetzko et al. [18] used ellipsoid pixel placement for the same purposes. Staib et al. [19] applied blurring to encode depth of field information. Such design choices for visual encoding can significantly enhance or degrade their visual quality. Micallef et al. [2] proposed a cost function aiming at an automatic optimization of marker size and opacity, aspect ratio, color, and rendering order in scatterplots depending on the analysis task.

Complex and large data cannot be effectively represented in traditional scatterplots due to *overplotting*. For overpopulated plots, there is a need to accentuate the most important data structures, reject or combine less relevant samples, or reduce the local sample density by other means. Recently, Sarikaya and Gleicher [20] surveyed existing scatterplot designs reasoned by the analysis tasks and data characteristics. A taxonomy of existing methods was developed by Ellis and Dix [21]. The authors identified three main strategies for tackling the overplotting issue, which can be characterized as spatial transformation, changing the appearance, and using animations.

Animations may help to encode temporally varying data or data with uncertainty [22]. Chen et al. [23] used flickering points for revealing multi-class structures in overplotted scatterplots. Technically, animations can handle relatively large datasets. However, one should consider the time required to show and perceive all the data as well as the cognitive burden [21].

Appearance change for clutter reduction may use different sizes for depicting objects [24]. More commonly applied though is the concept of opacity adjustment, which can improve the user's perception of local sample density in overplotted scatterplots. Matejka et al. [25] proposed a model for user-driven opacity scaling. Proper sampling of data can also improve the readability of visualization when the data size is large. Bertini et al. [26] performed a non-uniform sampling in combination with sample displacement to support user perception of scatterplots. *Splat*terplots proposed by Mayorga and Gleicher [27] alleviate the overdraw issue in traditional scatterplots by abstracting local sample density and rendering density contours. Hao et al. [28] split the scatterplot domain into bins along each spatial dimension and distribute data points within each bin, making individual samples accessible. Paulovich and Minghim [29] applied a hierarchical approach for visualizing document collection datasets. The proposed technique *HiPP* depicts data at a certain level of detail preserving the similarity of samples and their clusters as inter-object distances. Among these approaches that change the appearance of the scatterplots, opacity adjustment seems to be the most commonly applied method due to being simple, effective, and not introducing new visual objects. In our user study, we thus compare our method against opacity adjustment, see Section 6.3.

Spatial transformations for reducing clutter in scatterplots introduce a distortion of the sample placement. In the distorted view, high-density regions should be expanded, while low-density regions should be contracted to use the available screen space effectively. A taxonomy of distortion-oriented techniques was presented by Leung and Apper-

ley [30]. Sarkar et al. [31] used a rubber-sheet metaphor for a user-controlled local deformation of the visual domain. Keim et al. [32] proposed *generalized scatterplots* trading off overlap and distortion errors. The method combines a linear domain distortion technique [33] with a pixel displacement interactively controlled by the user. Recently, Raidou et al. [34] computed a space-filling transformation that maps data samples to free pixels. Vollmer and Döllner [35] proposed a collision detection algorithm and 2.5D layout for alleviating occlusion. Cutura et al. [36], [37] resolved collisions on space-filling curves for gridifying the scatterplot layout. Hilasaca et al. [38] remove overlapping of glyphs by creating dummy points. Liu et al. [39] applied constrained MDS for placing data items on a grid.

Local reduction of the samples' occlusion can be achieved by using *virtual lenses* or other similar *focus + context* techniques. *Fisheye views* proposed by Furnas [40] provide a smooth integration of two levels of details. *JellyLenses* developed by Pindat et al. [41] dynamically adapt their geometry to the content in the focus. The application of virtual lenses to scatterplots and parallel coordinates plots was discussed by Ellis et al. [42], [43]. Tominski et al. [44] presented a taxonomy of virtual interactive lenses according to the types of data and user tasks. Local reduction methods come with the drawback of requiring rather intense user interactions when exploring the entire visual space and imposing some cognitive load on the user when trying to compare different regions to each other.

In our work, we propose a novel global domain deformation technique, which aims at reducing the scatterplot occlusion. The deformation map computation is based on *summed-area tables* or *integral images* (InImms) originally introduced by Crow [45]. Viola and Jones [46] applied InImms to object detection in image analysis. Ehsan et al. [47] addressed the problem of efficient parallel computation of InImms. Singhal et al. [48] discussed the calculation of InImms on the embedded GPU using an OpenGL shader model. The proposed implementation uses a multi-pass 2D reduction technique performed in a fragment shader. Nowadays, many libraries like OpenCV [49] provide out-of-the-box functionality for efficient InIm computation. Reinbold and Westermann [50] presented a hierarchical approach for computing summed volume tables for sparse volumes. An extension to the classical InImms, which accumulate pixel values in the left-top corner of the input texture, are InImms tilted by 45°. Lienhart et al. [51], [52] presented a computation of tilted InImms on the CPU in two passes over all pixels. The authors used rotated InImms for the calculation of additional Haar-like features serving for a more robust detection of objects. Barczak et al. [53] extended the approach for 26.5° and 63.5° angles of rotation as well as constructed approximations for arbitrary angles. Computation of InImms at arbitrary angles was studied by Chin et al. [54].

Molchanov and Linsen [7] used InImms for computing virtual lenses and pseudo-cartograms. Their deformation formula cannot be used to achieve a nearly uniform distribution of samples in general cases, since the ideal regular distribution of samples is not a fixed point of the transformation. We correct the mapping presented in [7], so that an iterative regularization of arbitrary layout converges to the desired state, is stable, avoids overlapping of mapped

subareas, and has better performance scalability in terms of data size.

Spatial distortion changes the original distances between visualized elements, which may lead to misinterpretations of the data's structure by the user. Therefore, it is important to *inform the user about applied distortions* and so *relate the regularized layout back to its original representation*. Carpendale et al. [55] presented information about the properties of local map transformations via superimposed regular grids. In our work, we introduce different techniques for depicting spatial distortion including grids and color encoding of the background, which we evaluate within a user study in Section 6.3.

3 BACKGROUND

In this section, we provide basic concepts and necessary technical details of the existing algorithms, which we improve, adapt, and experiment with in our work. We start with a given real-valued density texture $d(i, j)$, whose construction is application-specific. The density depicts local weights, which may be interpreted as the amount of information or level of importance of the current locus. We propose a proper calculation of $d(i, j)$ for de-cluttering scatterplots later in Section 4.1. For easier reference to prior work on InImms (e.g., [7]), we use the same notations, where applicable.

3.1 Integral Images

For a given real-valued texture $d(i, j)$, an *InIm* is another real-valued texture $\alpha(i, j)$ of the same resolution, which is computed by summing up all values of the input texture $d(i, j)$ over its top-left corner up to the position of pixel (i, j) . We will also need a set of further sums of pixel values computed over the bottom-left, bottom-right, and top-right corners of the visualization domain, i.e.,

$$\begin{aligned}\alpha(i, j) &= \sum_{i' \leq i} \sum_{j' \leq j} d(i', j'), & \beta(i, j) &= \sum_{i' \leq i} \sum_{j' > j} d(i', j') \\ \gamma(i, j) &= \sum_{i' > i} \sum_{j' > j} d(i', j'), & \delta(i, j) &= \sum_{i' > i} \sum_{j' \leq j} d(i', j').\end{aligned}$$

Note that the areas of summations form a partition of the texture domain, i.e., they do not intersect and their union is the complete texture domain. Thus, $\alpha + \beta + \gamma + \delta \equiv C$ holds for all pixels, where $C = \sum d(i, j)$ is the sum of all pixel values of the given texture.

Next, we define *tilted InImms* with a tilt of 45° by

$$\begin{aligned}\alpha_t(i, j) &= \sum_{\substack{i'+j' \leq i+j \\ i'-j' \geq i-j}} d(i', j'), & \beta_t(i, j) &= \sum_{\substack{i'+j' \leq i+j \\ i'-j' < i-j}} d(i', j'), \\ \gamma_t(i, j) &= \sum_{\substack{i'+j' > i+j \\ i'-j' < i-j}} d(i', j'), & \delta_t(i, j) &= \sum_{\substack{i'+j' > i+j \\ i'-j' \geq i-j}} d(i', j').\end{aligned}$$

Analogously to InImms, the summation areas form a partition of the domain, and the equation $\alpha_t + \beta_t + \gamma_t + \delta_t \equiv C$ holds. InImms α and α_t can be computed using standard functions provided by many image processing libraries. The other tables, i.e., β, γ, δ , and their tilted versions, can be computed as standard InImms of the density texture when rotated by 90°, 180°, and 270°, correspondingly.

Thus, taking into account these two relations between the introduced integral tables, there are six linearly independent textures, namely $\alpha, \beta, \gamma, \alpha_t, \beta_t$, and γ_t . InLms provide a pixel-centered description of the density distribution on the global scope. Based on their values, it is possible to globally define a density-equalizing mapping, which automatically preserves the ordering of samples.

3.2 Domain Transformation

Texture $d(i, j)$ represents density values. High values indicate that more visual space is required locally for optimal representation, i.e., the visualization domain should expand locally. Low-density values identify regions that may be contracted to free more space. However, the density texture itself does not show preferable directions of deformation due to the lack of global scope.

InLms provide integrals of the density distribution over respective domains. Thus, their values represent globally aggregated characteristics of density distribution. Such information can be used for determining the direction and magnitude of local transformation leading to the globally improved (i.e., more uniform) distribution.

Molchanov and Linsen [7] proposed a global mapping of the following form:

$$\begin{aligned} t(x, y; d) = & (\alpha \cdot q_1(x, y) + \beta \cdot q_2(x, y) + \\ & \gamma \cdot q_3(x, y) + \delta \cdot q_4(x, y) + \\ & \alpha_t \cdot (x, 1) + \beta_t \cdot (1, y) + \\ & \gamma_t \cdot (x, 0) + \delta_t \cdot (0, y)) / (2C). \end{aligned} \quad (1)$$

Here, for each pixel with index (i, j) corresponding to the texture coordinates $(x, y) = 2^{-k}(i, j)$ for a texture of resolution $2^k \times 2^k$, four anchor points $q_l(x, y)$, $l = 1, \dots, 4$ are defined as follows:

$$\begin{aligned} y < x : \quad & q_1(x, y) = (1, 1 + y - x), \\ & q_3(x, y) = (x - y, 0); \\ y \geq x : \quad & q_1(x, y) = (1 - y + x, 1), \\ & q_3(x, y) = (0, y - x); \\ x + y < 1 : \quad & q_2(x, y) = (x + y, 0), \\ & q_4(x, y) = (0, x + y); \\ x + y \geq 1 : \quad & q_2(x, y) = (1, x + y - 1), \\ & q_4(x, y) = (x + y - 1, 1). \end{aligned}$$

For completeness and self-sufficiency of the exposition, we reproduce Fig. 1 from Molchanov and Linsen [7], where the anchor points' positions and important notations are depicted, see Figure 1.

Molchanov and Linsen used mapping 1 for computing a user-steered virtual lens. The user selects a (multi-component) area of interest in the visual domain. Then, a density function taking constant values inside and outside of the area of interest is constructed. The user adjusts the density value inside the selected area to achieve a desired level of magnification of the region after applying mapping 1. Optimal values for the density strongly depend on the shape and position of the area of interest and cannot be computed a priori. Moreover, mapping 1 is not an identity transformation for a constant texture $d(i, j)$ and therefore

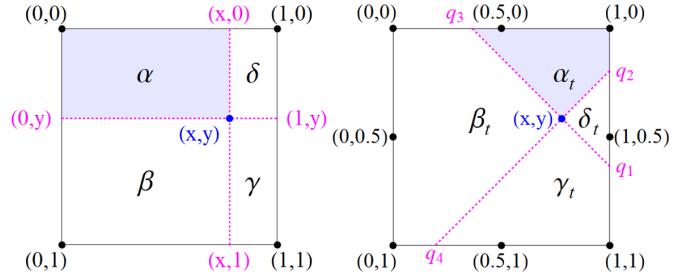


Fig. 1. Reproduction from Molchanov and Linsen [7]. Left: The four Inlm coefficients computed at location (x, y) stand for integrals of a density function over respective rectangular regions. Right: The four additional coefficients can be computed for the same location as integrals over tilted regions.

multiple applications of the mapping to the given distribution may lead to unpredictable results. Therefore, equalizing the distribution using mapping 1 is not possible.

4 DE-CLUTTERING SCATTERPLOTS

Given a scatterplot, our goal is to achieve a uniform distribution of samples in a fully automatic manner. We aim for stable and fast calculations that result in nearly uniform distributions for arbitrary initial configurations. Our first step is to construct a proper density function representing the given scatterplot (Section 4.1). Then, we modify mapping 1 to act as a converging density-equalizing transformation and solve the stability issue due to its potential singular behavior in the empty regions of scatterplots (Section 4.2). We further propose a computationally efficient implementation of the algorithm summarized in Section 4.3 on the GPU to allow its application within interactive data analysis and exploration systems (Section 4.4).

4.1 Density Field

Given a 2D scatterplot with n data samples $\mathbf{z}_i = (x_i, y_i)$, $i = 1, \dots, n$. Without loss of generality, we assume the set of samples belonging to the unit square, i.e., $\{\mathbf{z}_i\}_i \subset [0, 1]^2$. When rendering the scatterplot, the unit square is mapped to a discrete texture and shown on the screen. We assume that the texture has resolution $2^k \times 2^k$, $k \in \mathbb{N}$. For convenience, we place the origin in the top-left corner of the domain.

Given the distribution of samples $\{\mathbf{z}_i\}_i$ in the scatterplot, our goal is to define a deformation of the texture space $[0, 1]^2$, which would result in a more uniform distribution of the samples. The deformation has to be smooth to preserve neighborhood relations between samples, i.e., to avoid mixing and changing the local samples' order.

The first step of the proposed algorithm is the computation of a scalar-valued function, which describes the distribution of data samples in the scatterplot domain. We require this density function to be smooth since it serves as a basis for constructing the deformation map. Such a smooth density distribution can be computed by

$$d_r(x, y) = \sum_{p=1}^n \phi_r(x - x_p, y - y_p),$$

where ϕ_r is a smooth radial basis function, e.g., a 2D Gaussian kernel, and r is its dilation parameter. Smaller values of r correspond to more localized contributions from individual samples, while larger values of r result in a smoother density distribution.

In practice, one computes a discrete version in the form of the rasterized density $d_r(i, j)$ at pixels (i, j) . This can be performed using various techniques. One approach is an accumulation of splats in a single-channel float-valued texture as proposed for the construction of continuous scatterplots by Bachthaler and Weiskopf [56]. Then, larger values of r result in more blurred distributions and longer computational times due to the need to update a higher number of pixel values. Molchanov et al. [57] proposed using a spectral algorithm for a fast accumulation of large splats. In our application scenario, we use perfectly isotropic kernels with equal radii, since we assume no variability among the samples. Therefore, we can use a more efficient procedure. The most efficient method for computing density $d_r(i, j)$ is to restrict the contribution of each data sample to a single pixel of the texture. Then, we convolve the resulting texture with a discrete smoothing kernel such as the 2D Gaussian kernel, and accumulate the smoothed textures.

Depending on the original data and the value of parameter r , density texture $d_r(i, j)$ may contain vanishing or very small pixel values. When regularizing scattered data, empty regions should be contracted and ideally should disappear. The resulting mapping becomes singular in such regions, which can lead to numerical instabilities and some overlap of mapped subregions. Therefore, to ensure the stability of calculations, we add a global constant value to $d_r(i, j)$ for all pixels (i, j) . To summarize, the resulting density texture is then computed by

$$d(i, j) = d_r(i, j) + d_0. \quad (2)$$

Although the additive constant value d_0 could be any, it should not be too small relative to the maximum density, i.e., d_0 should depend on the number of samples n and the total number of pixels n_p in the texture. The average number of samples per pixel n/n_p is the theoretical density of the perfect uniform distribution. We used this constant in our experiments, which always worked well.

4.2 Smooth Global Deformation

Mapping 1 proposed by Molchanov and Linsen [7] enlarges the selected region of interest when the density value assigned to the interior part of the selection is larger than the background density. However, when both density values are equal, i.e., the density is constant over the entire visualization domain, mapping 1 destroys the uniformity of the distribution. Thus, it cannot serve as a density-equalizing transformation.

We fix this issue by computing the defect $t(x, y; d_0)$, i.e., the distortion mapping of the constant density texture of value d_0 , and subtracting it from transformation 1. Then, the transformation

$$t(x, y) = (x, y) + t(x, y; d) - t(x, y; d_0) \quad (3)$$

is an identity mapping for constant density textures: When samples are distributed evenly, $d = d_0$ and $t(x, y) = (x, y)$.

This adjustment allows for iterative application of the mapping to scatterplot data such that the iterative process converges towards a nearly uniform data distribution. Convergence of the proposed iterative regularization is demonstrated in our numerical tests in Section 6. Using transformation 3, all pixels of the texture are mapped to new positions. To compute the new positions of data samples \mathbf{z}_i , we perform a bi-linear interpolation using the mappings of the four closest pixels.

4.3 Algorithm

The proposed iterative algorithm consists of the following steps:

- 1) For a given set of 2D samples \mathbf{z}_i , we generate a smooth rasterized density function representing their distribution in the scatterplot domain. We generate it by summing up per-pixel sample contributions and storing the sum in an accumulation texture. Then, we apply a convolution operator corresponding to a smooth kernel with control parameter r , which results in a smooth density plot $d(i, j)$.
- 2) For the generated discrete density, we compute all ordinary and tilted InLms involved in the computation of mapping 3. Note that the defect mapping $t(x, y; d_0)$ in 3 does not depend on d . Therefore, $t(x, y; d_0)$ is computed only once and then reused in each iteration of the algorithm.
- 3) Transformation 3 determines images of texture pixels. The per-pixel mapping is approximated at the samples' positions using a bi-linear interpolation formula. New positions of the samples are then computed accordingly.
- 4) Steps 1-3 are repeated for the new sample distribution. The stopping criterion can be defined by the user depending on the application domain and analysis task. For instance, the user may want to terminate the regularization process after a fixed number of iterations, after a given elapsed computational time, or based on any measure indicating how regular the achieved distribution is or how significant the impact of the last iteration was.

The presented algorithm has linear complexity with respect to the number of samples. Thus, the primary application domain of the method is the interactive exploration of large datasets.

4.4 Efficient GPU Computations

We implemented crucial steps of the proposed algorithm on the GPU. In particular, we developed a novel scheme to efficiently compute (tilted) InLms (Step 2).

In Step 1 of our algorithm, the subroutine accumulates per-pixel contributions of 2D samples in a high-resolution texture and applies a smoothing procedure. The accumulation is done using vertex and fragment shaders with additive blending. When smoothing is performed using a separable kernel, e.g., the Gaussian kernel used in our experiments, the convolution can be performed with two compute-shader passes (vertical and horizontal directions separately).

In Step 2, we compute the InImms involved in the constructions of the mapping. Singhal et al. [48] described computation of InImms on the GPU using a 2D reduction technique performed in the fragment shader. We adopt this algorithm for efficient computation of InImms and extend it to tilted InImms, see Figure 2. (i) For a given scalar-valued density texture, we first compute upper- (red) and lower-column integrals (green) as shown in Figures 2a–2c. The computations are progressive [58], where the summed area doubles in each step. For a square texture of size $2^k \times 2^k$, the procedure requires one rendering pass with k summation operations to complete. Our implementation launches one workgroup for each row/column/diagonal in the image. Each thread only relies on data computed within the same workgroup. As such, we only need a single rendering pass with a for-loop of k iterations and a barrier synchronizing the work group after each iteration. The resulting two-channel texture contains sums of density values at the pixels located above and below the considered pixel, respectively. (ii) The *classical InImms* can then be found in an analogous multi-pass procedure by collecting the computed column-integral values horizontally as shown in Figures 2d–2f. (iii) For the computation of tilted InImms, we precompute four *auxiliary triangle integrals* as presented in Figures 2g–2i. Their calculation is based on summing up the column integrals along respective diagonals. (iv) Finally, the *tilted InImms* can be easily computed by performing arithmetic operations on the auxiliary triangle textures, see Figures 2j–2l for α_t . In the figure, the upper-left and upper-right auxiliary triangle integrals are summed and the twice-counted column integral is subtracted. The other three tilted integrals β_t , γ_t , and δ_t can be computed analogously.

In Step 3, the two-channel deformation texture $t(x, y)$ can then be calculated according to 3 at every pixel. Bilinear interpolation of the deformation texture $t(x, y)$ at the sample positions results in a new set of mapped 2D samples. The dependence of the execution time on the algorithm's parameters and the data size is extensively explored in the numerical tests presented in Section 6.

5 VISUAL ENCODING OF LOCAL DEFORMATION

Structures and patterns (such as clusters, outliers, dense and sparse regions) of the samples' distribution in scatterplots may reveal features and characteristics of the studied phenomena. Occlusion hinders managing and accessing the samples when the data size is large. Our approach mitigates the overplotting issue by regularizing the samples' distribution in the scatterplot. A regularized layout is by definition patternless. Though the regularized layout is not designed to completely replace the original configuration, preserving the main data features of the original scatterplot could be beneficial. Recognition of patterns of the original scatterplot after deformation motivates us to explore visual encodings, which could help to convey data structures, at least partially. The approaches discussed in this section are evaluated in the user study of Section 6.3 (cf. Task T3).

Our mapping 3 is an affine combination of points on the domain boundary with non-negative weights that vary smoothly. Moreover, since the InImms are by construction monotonic, i.e., they have no extreme points in the interior

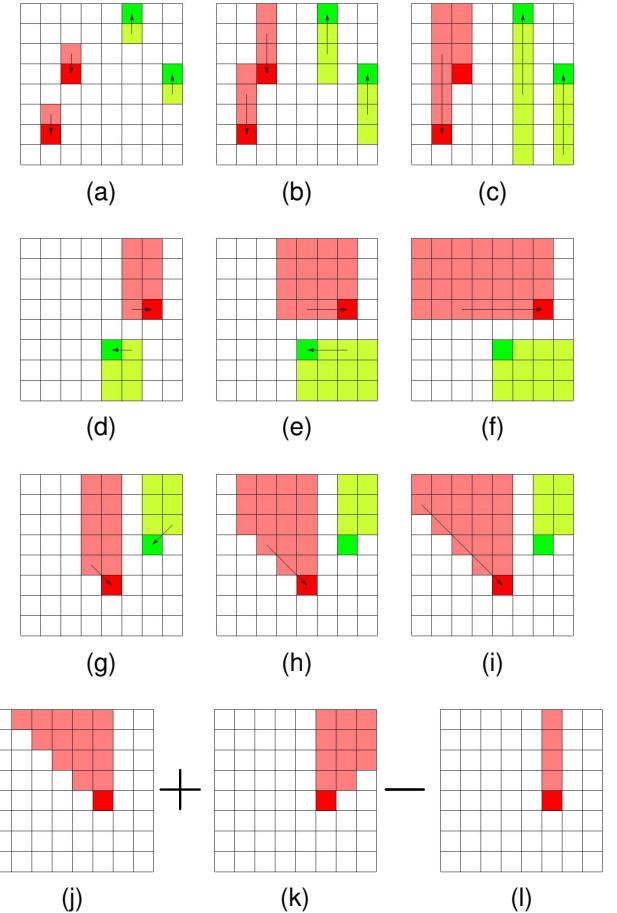


Fig. 2. Efficient computation of InImms. (a)–(c) Computation of column integrals. Pixels highlighted in red iteratively accumulate texture values from the pixels located above them. Green pixels progressively sum up values from the pixel columns below. After k summations are performed in a single rendering pass, every pixel contains upper- and lower-column sums of values stored in two texture channels. (d)–(f) Computation of InImms by iterative accumulation of column integrals. In the last iteration, red and green pixels contain values for α and γ , correspondingly. InImms for β and δ can be computed analogously. Note that β , γ , and δ can be evaluated on demand using α only, thus, their explicit computation is not necessary. (g)–(i) Calculation of triangle integrals by summing up column integrals along diagonals. Two of the four required auxiliary integrals are shown. (j)–(l) Tilted InImms can be computed by simple arithmetic operations on precomputed column and triangle integrals. An example for calculating α_t is presented. Tilted InImms β_t , γ_t , and δ_t can be found analogously.

of the domain, the resulting mapping also preserves some degree of monotonicity. In particular, it does not introduce any discontinuities or twisting/swirling behavior. Thus, neighborhood relations of samples are generally preserved, meaning that points in the neighborhood of a point will remain in the neighborhood after deformation. However, this does not mean that the k nearest neighbors will remain the k nearest neighbors, as distances between points will, in general, be distorted anisotropically. Preservation of neighborhoods is important since the regularized layout facilitates accessing and managing samples. Preservation of other characteristics of the undistorted distribution is per se not the goal of the de-cluttering approach, but could nevertheless be beneficial for the user. Indeed, since distances between samples are relaxed, some data analysis tasks can-

not be reliably performed on the regularized plot. However, additional visual cues may make data analysis possible even without switching back to the original scatterplot layout.

Identifying data clusters and outliers in the distorted map requires the analyst to take into account the local characteristics of the applied transformation. Such information needs to be visually conveyed. We propose to use different visual encodings of the local deformations and evaluate them in a user study.

As a first option, we follow the approach by Carpendale et al. [55] and draw a *sparse regular grid*, whose nodes we deform just like the data samples by using transformation 3. Based on the local density of the grid lines after deformation, one can judge the local area expansion or contraction factor as well as its direction. The method is simple and has been applied successfully in other studies. However, it introduces additional geometry, which increases visual complexity and thus adds to the occlusion in the plot.

As a second option, we propose to use a *density background texture*. The density field of the given scatterplot is computed as described in Section 4.1. It is then deformed using our mesh mapping from Section 4.2. Finally, we apply a transfer function to map densities to colors and render the resulting texture as the background of our deformed scatterplot. Consequently, we expect high-density areas such as data clusters to remain visible after distortion. More precisely, high-intensity regions denote the cores of the clusters and the low-intensity regions depict the separation of clusters. Alternatively, when the background texture is used for other purposes, one may use a *density color-coding of the samples*. Thus, one encodes the original density at the positions of the transformed samples using color. This option follows the ideas that are also used in appearance change for clutter reduction, see Section 2. We therefore leave this alternative out of the scope of this paper.

Finally, density variation within original data clusters can be roughly depicted using contour lines of the density function. Showing the same contour lines after density-equalizing transformation allows for identifying the clusters' boundaries. Figure 3 compares visual encodings of the applied transform using grid lines, a density texture, and contour lines after regularization of a UMAP embedding of the MNIST dataset [59]. Figure 3c and Figure 3d reveal subcluster structures that could not be inspected in the original scatterplot due to high occlusion. Results of the user study comparing the benefits and effectiveness of different approaches are presented in Section 6.3.

6 RESULTS

We first present some results that visually show the de-cluttering process on various scatterplot examples (Section 6.1), then perform some numerical tests on our algorithm in terms of performance, quality, and stability (Section 6.2), and finally describe the user study we conducted (Section 6.3).

6.1 Visual Investigations

In the first experiment, we illustrate the regularization effect of the proposed algorithm. We performed a few iterations

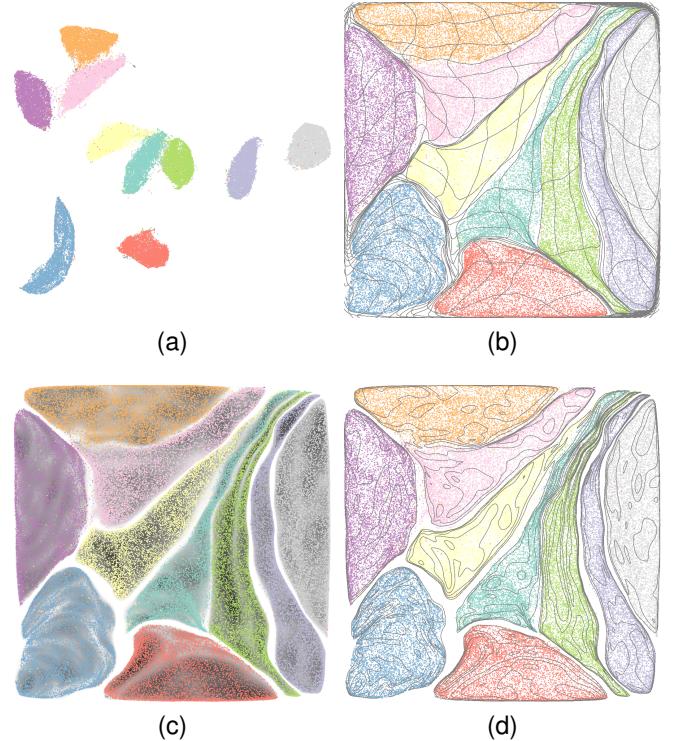


Fig. 3. (a) Original layout of the MNIST dataset (UMAP, number of neighbors 15, minimal distance 0.1) with color-coded classes. (b) Visual encoding of the density-equalizing transform using grid lines after 32 iterations. The original density of samples is represented by the background texture in (c) and by contour lines in (d). The last two figures allow for analyzing the subcluster structures occluded in (a).

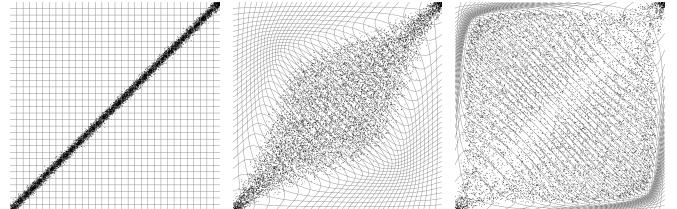


Fig. 4. Regularization of data sampled roughly along the domain diagonal. A superimposed regular grid conveys the domain deformation. Left: Original scatterplot. Middle: After 2 iterations. Right: After 8 iterations.

starting with samples distributed roughly along the domain diagonal (Figure 6.1). The deformation mapping is conveyed by showing a transformation of a regular grid throughout the iterations. When samples are placed along the diagonal, distributions of the data in x and y dimensions are uniform. Therefore, the application of the HistoScale-based approach proposed by Keim et al. [33] would have no effect. Our algorithm efficiently deforms the visual domain spreading the samples from the diagonal over the available void space.

In a second experiment, we demonstrate that no mixing of samples from different clusters takes place during the deformation presented in Figure 5. During iterative transformations, the four clusters significantly change their shape such that the free space between the clusters is contracted. Still, due to the smooth and monotonic character of the transformation, no samples from one cluster can be mapped

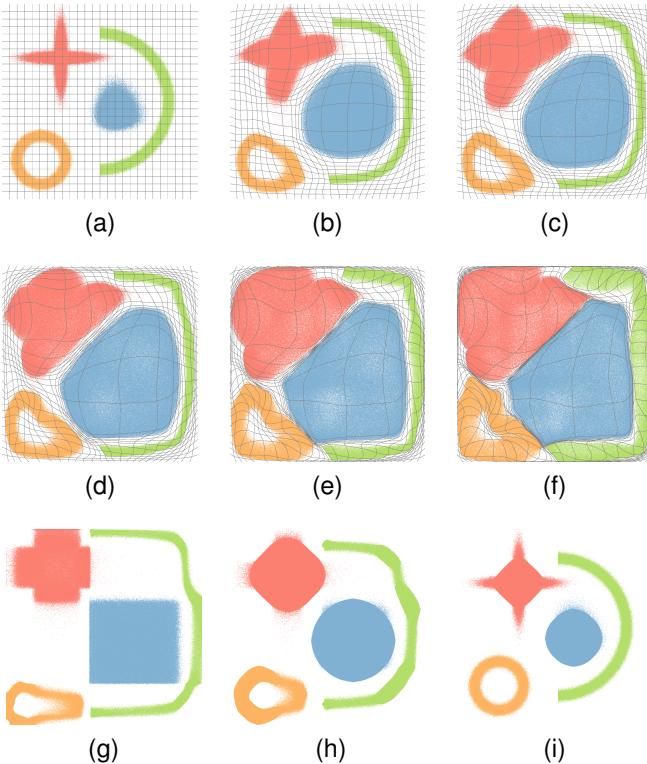


Fig. 5. Regularization of samples' distribution in scatterplot. (a) Original scatterplot depicts four clusters shown in blue (400k samples), red (300k samples), green (200k samples), and orange (100k samples). Visual estimation of cluster sizes as well as access to individual samples are hindered by excessive overplotting. (b)-(f) Iterative transformation of scatterplots using the proposed de-cluttering algorithm after 1 (b), 2 (c), 4 (d), 8 (e), and 16 (f) iterations. Computational times are 1.15 ms, 2.99 ms, 6.22 ms, 12.76 ms, and 25.24 ms respectively. After a few iterations, data clusters occupy areas proportional to the number of samples contained in them. No mixing of clusters takes place. A superimposed regular grid is deformed using the same mapping. The shape of the deformed grid represents the computed mapping and may serve for the identification of the original data clusters even if they were not color-coded. (g)-(i) Generalized Scatterplots proposed by Keim et al. [32] demonstrate noticeably less efficient use of the screen space for any combination of the governing parameters: (g) distortion = 1, overlap = 0.1, (h) distortion = 0.5, overlap = 0.05, (i) distortion = 0, overlap = 0.1.

to the area occupied by the samples from the other cluster. All clusters remain separated by thin unoccupied areas distinguished by a high density of grid lines. Figure 6 shows a data set with three clusters, where within each cluster regions of different shapes are manually selected in the original scatterplot (selections are highlighted by color). We observe that the selection boundaries remain sharp after regularization. Thus, even closely related samples within the same cluster preserve their local order. Yet another experiment demonstrating the preservation of the samples' neighborhoods is shown in the accompanying video.

For some applications, it is desirable to replace point renderings in scatterplots with glyphs or icons, which require more space. Figure 7 shows a scatterplot (with background coloring) of an image data collection with image icons. De-cluttering of the original layout results in a significant reduction of the icons' overlap using an efficient use of the available screen space. Therefore, the user can better inspect

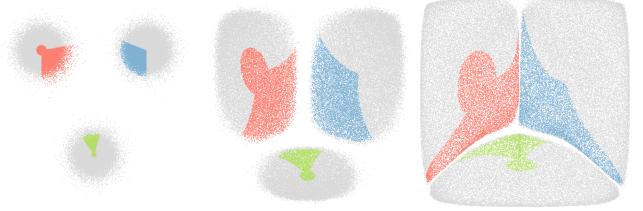


Fig. 6. De-cluttering scatterplot with manual selections of different shapes highlighted by color. Selected regions change their geometry during the regularization iterations but preserve their sharp boundaries. No mixing of highlighted and other samples occurs. Left: Original scatterplot. Middle: After 2 iterations. Right: After 16 iterations.

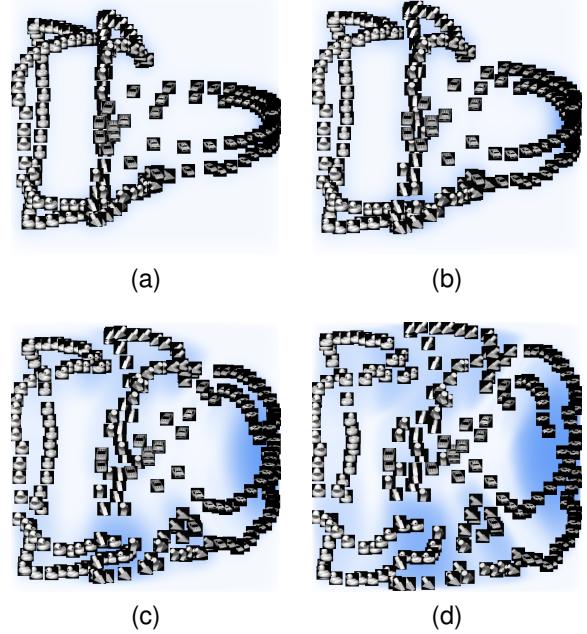


Fig. 7. De-cluttering scatterplot of COIL dataset [60] with image icons. After a few iterations of our regularization, the user has a better overview of the variability of the data within clusters and can more easily access individual samples, as the screen space is used more effectively. The background texture encodes the original density of samples to reveal cluster structures after deformation. (a) Original, 2.92 ms. (b) 2 iterations, 4.36 ms. (c) 8 iterations, 13.50 ms. (d) 16 iterations, 26.37 ms.

the variability of samples within clusters, access individual samples, estimate the density and numbers of data samples, and analyze class-cluster relations.

6.2 Numerical Tests

Performance and quality measures. Table 1 shows that the *computation time* scales linearly with the number of points and the number of iterations performed. Computation times are small enough for embedding our approach into interactive visual systems, even for large data sets.

To judge the *quality of our regularization*, we split the domain into bins of size 4×4 pixels and compute the number of samples in each of these bins. Then, the standard deviation of the computed values from the mean number of samples per bin can serve as a regularization measure. The standard deviation vanishes for a perfectly regular sample distribution. Results presented in Figure 8, left show

TABLE 1

Computation times on a GeForce RTX 2060 for a texture size of 1024×1024 and a kernel size of $r = 8$.

iter.	500k	1M	2M	4M
1	1.16 ms	1.44 ms	1.98 ms	3.07 ms
2	2.33 ms	2.95 ms	4.12 ms	6.40 ms
3	3.56 ms	4.51 ms	6.33 ms	9.89 ms
4	4.78 ms	6.06 ms	8.53 ms	13.33 ms
5	6.06 ms	7.66 ms	10.81 ms	17.05 ms
6	7.34 ms	9.30 ms	13.16 ms	20.81 ms
7	8.63 ms	10.96 ms	15.56 ms	24.67 ms
8	9.92 ms	12.63 ms	17.97 ms	28.55 ms

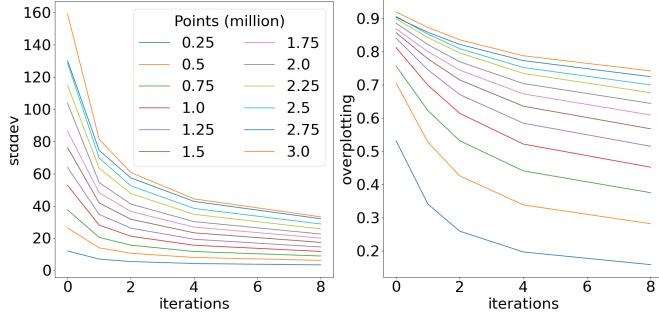


Fig. 8. Quality and performance measures of the proposed de-cluttering algorithm. The lowest curve (blue) corresponds to $250k$ samples. For each consecutive curve, the number of points is increased by $250k$ such that the upper-most curve (orange) corresponds to $3M$ points. Standard deviation from the mean number of samples and overplotting decrease monotonically, i.e., the samples' density becomes uniform.

a monotonic decrease of the observed quantity over the iterations, i.e., the sample distribution becomes more and more regular.

We furthermore compute the amount of overplotting as the difference between the total number of samples and the number of occupied pixels divided by the total number of samples. Figure 8, right shows that overplotting monotonically decreases throughout the iterative de-cluttering. To compute binning and overplotting measures, we generated 500 random datasets containing 1 to 8 variable-sized Gaussian clusters at random locations similar to [27].

Comparison with the state of the art. *Generalized Scatterplots* proposed by Keim et al. [32] use linear distortions in horizontal and vertical directions similar to the *HistoScale* method [33] enhanced with a pixel-placement procedure. The linear distortion is the most similar technique to our method. Figure 4 shows an example of a data set that cannot be de-cluttered using the *HistoScale* approach. Thus, our regularization algorithm performs significantly better than the axis-aligned transformation of *HistoScale*. Figure 5 shows a visual comparison of our approach and the *Generalized Scatterplots* (for different parameter settings). Our technique uses the available screen space significantly better, which reduces occlusion. Moreover, in contrast to *Generalized Scatterplots*, our approach avoids overlap of the mapped data and does not require setting up related parameters.

Opacity adjustment using α -blending is arguably the most popular technique for mitigating visual clutter in scatterplots. Since this approach falls into the category of appearance change instead of spatial transformation, an objective

comparison is less obvious. Therefore, we extensively compare our proposed regularization technique with opacity-adjusted scatterplots in our user study. Results are presented in Section 6.3.

We compared the results of the proposed de-cluttering algorithm with the state-of-the-art approach for relaxed scatterplots developed by Raidou et al. [34]. We used the same datasets as the authors and refer the reader to the supplementary material for details on the produced visualizations. As datasets A through H were not available to us, we recreated them as closely as possible based on the information in their paper. First, we observe that, if the domain resolution is fixed, the pixel-based mapping proposed by Raidou et al. cannot handle datasets with the number of samples exceeding the number of pixels. Although the canvas size can be increased to satisfy the condition, a large canvas size may potentially exceed the maximal texture size on the GPU, which would hinder its use for fast computations. In our proposed approach, such a limitation does not exist: The texture size affects only the quality of the resulting regularization and can always be chosen to match the GPU characteristics. Datasets of arbitrary sizes can be processed since the point-based data is converted into a density distribution rasterized on the given canvas. Moreover, while we are preserving neighborhoods, the method by Raidou et al. does not guarantee any preservation of the neighborhood relations (see Datasets G and H in supplementary material), since it aims at minimizing displacements of samples. Overall their deformation pattern (displacement vector field) is much more irregular than the one in our approach. For example, close samples can be shifted to random directions, which is especially noticeable in low-density regions. Instead, our proposed algorithm preserved samples' local relations even in extreme cases, see Figure 9.

Another state-of-the-art overlap-reducing algorithm, *Hagrid*, was developed by Cutura et al. [36], [37]. The authors made the source code available for testing and evaluation. We compare our approach against *Hagrid* using a number of quality metrics proposed by Hilasaca et al. [38] for scatterplots with glyphs. Out of those metrics, *glyphs' overlap* and *layout spread* do not apply to scatterplots with no glyphs. *Stress* and *displacement* implicitly depend on the glyphs' size: When glyphs are small enough, short displacements are sufficient to remove overlaps. Therefore, it is not directly possible to apply these metrics to our proposed algorithm, which is designed to reproduce a globally uniform distribution of samples with no glyphs. *Aspect ratio* is trivially conserved in our application case. However, *trustworthiness* [61] and *orthogonal ordering* [62] evaluate the preservation of local neighborhoods and relative positions of regularized samples, which we visually confirmed for our proposed algorithm, e.g., in Figure 6. Additionally, we now present results for the mentioned quality metrics when applied to 2,832 scatterplot layouts generated using numerical attributes of datasets from UCI repository [63] (see supplementary material for technical details) in Figure 10. Our proposed algorithm (denoted as Inlm) demonstrates significantly better results in preserving local neighborhoods and relative positions of samples when compared to *Hagrid*. *Hagrid* tends to break these properties

when resolving collisions along the space-filling curves.

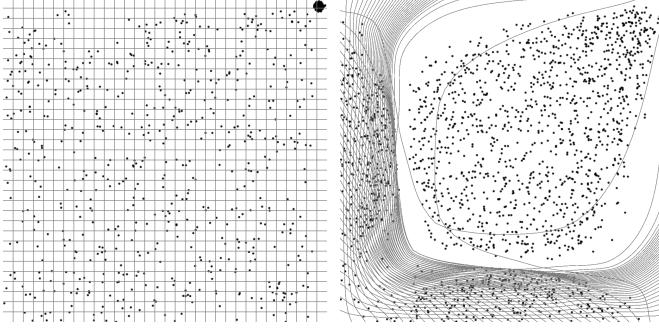


Fig. 9. De-cluttering of the recreated Dataset D from Raidou et al. [34] using our proposed algorithm. Left: Original scatterplot. Right: After 4 iterations.

The complexity of the pixel-based relaxation algorithm by Raidou et al. [34] is $\mathcal{O}(n^3)$, which limits its application to large datasets. Therefore, the authors proposed to use an approximating median-split point-to-pixel mapping, which reduces the complexity to $\mathcal{O}(n \log n)$ at the cost of worse preservation of the points' locality. The complexity of the collision handling part of Hagrid algorithm [37] is $\mathcal{O}(n^2)$ (with $\mathcal{O}(n \log n)$ operations in the best-case scenario). Our algorithm has a linear asymptotic complexity in the number of samples n plus $\mathcal{O}(m)$ operations performed on the density texture with $m = 2^k \times 2^k$ pixels to compute InLms, leading to an overall complexity of $\mathcal{O}(n+m)$. Table 1 shows interactive performance rates even for datasets two orders of magnitude larger than the biggest datasets used in [34] and [37].

Smoothing parameter and background density value. Smooth density fields ensure a smooth transformation of the scatterplot domain. The density smoothness is controlled by the kernel size r . Larger values of r result in more regular deformations as shown in Figure 11. Our experiments showed that the results are robust against small changes of r . A sufficiently large r can, therefore, be set as default and does not require manual parameter tuning.

Our tests showed that vanishing density can lead to overlapping of mapped regions. Very large kernel sizes could alleviate the issue. However, we instead proposed to use the additive constant density introduced in 2, which eliminates vanishing density regions and therefore effectively prevents overlapping regions. This is a better solution than using very large kernel sizes as large values of r lead to longer computation times as well as slightly slower convergence of the iterative regularization.

Parameter k defining the texture size should be large enough to achieve good sample separations in the regularized layout. However, unreasonably high values of k result in increased consumption of memory and massively worsen the performance.

6.3 User Study

We conducted a quantitative user study to evaluate our approach against classical scatterplots with opacity adjustment, which is arguably the most commonly used method for clutter reduction in scatterplots. The goal of the study

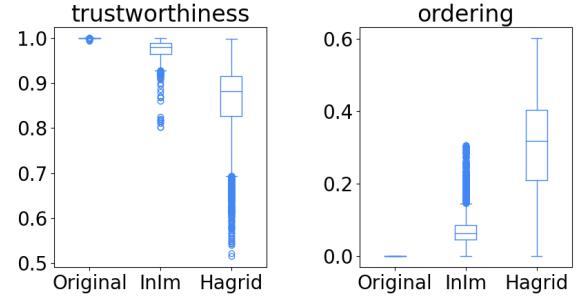


Fig. 10. Numerical comparison of our proposed method (Inlm) with Hagrid using 2,832 scatterplots. Quality metrics *trustworthiness* and *ordering* characterizing the preservation of local data structure are evaluated. Our algorithm (referred to as Inlm) results in significantly better values than Hagrid.

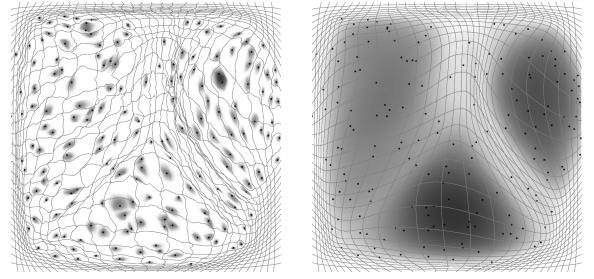


Fig. 11. Effect of varying size of the smoothing kernel on regularizing deformation for “Wine” dataset from UCI machine learning repository [63]. Left: Very small kernel size results in an insufficiently smooth density distribution, which leads to a low quality of the mapping. Wriggled grid lines indicate the bad character of the mapping. Right: Larger values of parameter r improve the smoothness of the density and result in a well-behaved regularizing transformation.

is to demonstrate that regularization may help to solve or mitigate issues related to occlusion (e.g., estimation of local number of samples, analysis of class-cluster interplay, accessibility of individual samples), while it is possible to overcome difficulties arising in the distorted layout (e.g., lack of clear cluster separation, equalized inter-sample distances) by suitable visual encodings of the distortion.

Setup. A total of 25 participants (aged 20 to 58 with an average of 28.56, 21 male, 3 female, 1 diverse) were recruited among friends and colleagues to perform three different tasks on six datasets each. For each dataset, we measured the participants' accuracy, speed, and confidence. For the first two tasks, we compared two visualizations: the original scatterplot with density encoding and our regularized scatterplot. For the third task, we compared three visualizations: the original scatterplot, our regularized scatterplot with a grid, and our regularized scatterplot with a background texture. The participants were randomly split into equally sized groups for each task. The study was conducted online and was fully anonymous. Before each task, the visualizations and interaction mechanisms were explained in detail. All participants were, generally, familiar with scatterplots, but not all of them were visualization experts. Therefore, participants could familiarize themselves with the effects of our regularization approach on scatterplots and the different visual encodings by interactively changing the number of

regularization iterations on an example scatterplot in a short training set-up. For the selection of points using a lasso selection tool, we had the participants perform a small test to make sure that interactions were correctly understood. Each session took approximately 25 minutes.

Datasets and tasks. All 18 datasets used for the user study are synthetic. Their visualizations are shown in the supplementary material. The participants were asked to complete the following three analysis tasks on the described datasets:

- T1 *Estimation of relative cluster sizes:* Given a scatterplot with two separate clusters, the participants were tasked to estimate what percentage of all points is in one of the clusters. The accuracy was measured using the absolute difference to the correct percentage. The clusters had the same shape to avoid unwanted visual effects that could affect the estimation of their sizes.
- T2 *Sorting of clusters:* Given a scatterplot with multiple colored clusters, the participants were tasked to sort them by their number of points. The accuracy was measured by the number of correct pairwise relative positions in the ordering. All clusters represent two-dimensional Gaussian distributions with different amounts of variance. Their positions in the scatterplot were randomized.
- T3 *Selection of clusters:* Given a scatterplot with multiple clusters, the participants were tasked to select the clusters using a provided lasso selection tool. The accuracy was measured by the percentage of correctly selected samples. This task included datasets with more complex cluster shapes. In particular, non-convex clusters such as arcs.

These tasks cover the three task categories *object-centric*, *browsing*, and *aggregate-level* for scatterplot designs proposed by Sarikaya and Gleicher [20]. More precisely, the tasks *object comparison* (4), *search for known motif* (6), *characterize distribution* (8), and *numerosity comparison* (11) are included.

Hypotheses. We formulate the following hypotheses:

- H1 Visual analysis of the data class-cluster composition is more precise when using our proposed approach compared to the classical scatterplots with density-based opacity adjustment (T2).
- H2 Estimation of the cluster size is more accurate when using our proposed algorithm compared to the classical scatterplots with density-based opacity adjustment (T1).
- H3 Detection and selection of clusters is equally accurate in classical scatterplots with density-based opacity adjustment and when using our regularization enhanced by visual encodings of local deformations (T3).
- H4 Visual encodings of local deformations lead to equal accuracy with deformed grids and background coloring (T3).

Statistical analysis. We tested the null hypothesis that all approaches perform equally well in accuracy, speed, and confidence. For the statistical analysis, we look at the p-value, calculated using a two-sample unpooled t-test, and the effect size, calculated using Cohen's d [64]. We consider

the p-value to be significant if it is smaller than the significance level of 0.05.

Results and discussion. All statistical information about our analysis is provided in the supplementary material. For the first task, the regularized visualization performed significantly better than the original scatterplot with opacity adjustment in both accuracy ($p = 0.008, d = -0.569$) and confidence ($p < 10^{-3}, d = 1.154$). We therefore accept H2. For the second task, the regularized visualization performed significantly better than the original scatterplot with opacity adjustment in both accuracy ($p < 10^{-3}, d = 0.713$) and confidence ($p < 10^{-3}, d = 0.752$). We therefore accept H1. For the third task, the regularized visualization with background texture performed significantly worse than the original scatterplot in accuracy ($p = 0.048, d = -0.400$) and the regularized visualization with grid performed significantly worse than the regularized visualization with background texture in accuracy ($p < 10^{-3}, d = -1.142$), speed ($p = 0.027, d = 0.452$), and confidence ($p < 10^{-3}, d = -1.053$). We therefore reject H3. We also reject H4, as the background texture visualization performed significantly better than the deformed grid. A limitation of our study is the exclusive use of Gaussian clusters for Task T2. More complex cluster shapes would have made the task more difficult for participants and possibly caused unintended side effects. Moreover, since our approach does not preserve the shape of clusters, we expect the same results for non-Gaussian clusters.

7 CONCLUSION

We proposed an algorithm for a data-driven deformation of the visual domain, which can be interactively toggled by the user to de-clutter the scatterplot layout. The algorithm is deterministic, computes progressive regularizations of the initial layout, and has theoretical and practical complexity of $\mathcal{O}(n + m)$ as it does not require any collision detection of samples. Thus the user can change the desired degree of regularization applied to large datasets in run time. An efficient GPU implementation is provided as an open-source code. The resulting density-equalized layouts preserve original local relations of data items and allow for better accessibility of samples, accurate selections, and easier analysis of class-cluster characteristics. Although some data analysis tasks are better performed in the original scatterplot, principal information about the data structure can be still visually presented in the de-cluttered layout, which can be beneficial for the user. We investigated several approaches for visual encoding of the deformation map including grid, density texture, and contour lines. While grid lines better characterize the resulting deformation, density texture helps to identify data clusters and contour lines reveal subcluster structures, which may be occluded in the original plot.

The choice of parameters (additive constant density, kernel size) is discussed and visually illustrated. Computational efficiency, preservation properties and application scenarios of the proposed technique were visually and numerically explored using about 3,400 different scatterplot layouts. We compared our method to four existing approaches [32]–[34], [36] by constructing representative examples and evaluating important quality metrics. Detailed

information about our tests is presented in the supplementary material and the accompanying video.

Most of the existing regularization approaches compute the displacement of each sample based on local data distribution. Frequently, it leads to collisions when several samples are moved to the same location. Such collisions should be then detected and resolved, making the overall algorithm complex and slow. In contrast, the proposed technique determines individual displacements of samples based on information about data distribution on the global scope. This information is encoded in a set of InLms, which can be efficiently computed. So, we ensure a monotonous decrease of samples' density in the whole spatial domain and avoid collisions, therefore achieving fast convergence to a nearly uniform sample distribution.

Density-equalizing mappings find their application in various visualization tasks. Future directions of research may include the application of the proposed technique to local lenses in scatterplots, general optimization of items' placement in layouts, and construction of contiguous cartograms.

ACKNOWLEDGMENTS

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) – MO 3050/2-3 and CRC 1450 – 431460824.

REFERENCES

- [1] D. Eler, M. Nakazaki, F. Paulovich, D. Santos, G. Andery, M. C. Oliveira, J. a. Neto, and R. Minghim, "Visual analysis of image collections," *The Visual Computer*, vol. 25, pp. 923–937, 10 2009.
- [2] L. Micallef, G. Palmas, A. Oulasvirta, and T. Weinkauf, "Towards perceptual optimization of the visual design of scatterplots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 23, no. 6, pp. 1588–1599, June 2017.
- [3] 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*, ser. AVI '02. New York, NY, USA: Association for Computing Machinery, 2002, pp. 167—176. [Online]. Available: <https://doi.org/10.1145/1556262.1556289>
- [4] R. Hu, T. Sha, O. Van Kaick, O. Deussen, and H. Huang, "Data sampling in multi-view and multi-class scatterplots via set cover optimization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 26, no. 1, pp. 739–748, 2020.
- [5] J. Yuan, S. Xiang, J. Xia, L. Yu, and S. Liu, "Evaluation of sampling methods for scatterplots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 2, pp. 1720–1730, 2021.
- [6] J. S. Yi, Y. a. Kang, J. Stasko, and J. A. Jacko, "Toward a deeper understanding of the role of interaction in information visualization," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1224–1231, 2007.
- [7] V. Molchanov and L. Linsen, "Smooth map deformation using integral images," *Journal of WSCG*, vol. 28, no. 1–2, pp. 18–26, July 2020.
- [8] M. Friendly and D. Denis, "The early origins and development of the scatterplot," *Journal of the History of the Behavioral Sciences*, vol. 41, pp. 103–130, Feb 2005.
- [9] I. T. Jolliffe, *Principal Component Analysis*. Springer-Verlag, 1986.
- [10] E. Kandogan, "Star coordinates: A multi-dimensional visualization technique with uniform treatment of dimensions," in *Proceedings of IEEE Information Visualization Symposium*, 2000, pp. 4–8.
- [11] ———, "Visualizing multi-dimensional clusters, trends, and outliers using star coordinates," in *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '01. New York, NY, USA: ACM, 2001, pp. 107–116. [Online]. Available: <http://doi.acm.org/10.1145/502512.502530>
- [12] J. Kruskal, "Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis," *Psychometrika*, vol. 29, no. 1, pp. 1–27, 1964.
- [13] J. B. Kruskal and M. Wish, *Multidimensional scaling*. Beverly Hills, California: Sage Publications, 1978.
- [14] L. van der Maaten and G. E. Hinton, "Visualizing high-dimensional data using t-SNE," *Journal of Machine Learning Research*, vol. 9, pp. 2579–2605, 2008.
- [15] L. McInnes, J. Healy, N. Saul, and L. Großberger, "UMAP: Uniform Manifold Approximation and Projection," *Journal of Open Source Software*, vol. 3, no. 29, p. 861, 2018. [Online]. Available: <https://doi.org/10.21105/joss.00861>
- [16] M. Sedlmair, T. Munzner, and M. Tory, "Empirical guidance on scatterplot and dimension reduction technique choices," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 12, pp. 2634–2643, Dec 2013.
- [17] Y.-H. Chan, C. D. Correa, and K.-L. Ma, "Flow-based scatterplots for sensitivity analysis," *2010 IEEE Symposium on Visual Analytics Science and Technology*, pp. 43–50, Oct 2010.
- [18] H. Janetzko, M. C. Hao, S. Mittelstadt, U. Dayal, and D. Keim, "Enhancing scatter plots using ellipsoid pixel placement and shading," in *2014 47th Hawaii International Conference on System Sciences*. Los Alamitos, CA, USA: IEEE Computer Society, Jan 2013, pp. 1522–1531. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/HICSS.2013.197>
- [19] J. Staib, S. Grottel, and S. Gumhold, "Enhancing scatterplots with multi-dimensional focal blur," *Computer Graphics Forum*, vol. 35, no. 3, pp. 11–20, 2016. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12877>
- [20] A. Sarikaya and M. Gleicher, "Scatterplots: Tasks, data, and designs," *IEEE Transactions on Visualization and Computer Graphics*, vol. 24, no. 1, pp. 402–412, Jan 2018.
- [21] G. Ellis and A. Dix, "A taxonomy of clutter reduction for information visualisation," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, no. 6, pp. 1216–1223, Nov 2007.
- [22] D. Feng, L. Kwock, Y. Lee, and R. Taylor, "Matching visual saliency to confidence in plots of uncertain data," *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 6, pp. 980–989, 2010.
- [23] 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 *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, ser. CHI '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 1–12. [Online]. Available: <https://doi.org/10.1145/3173574.3173991>
- [24] J. Li, J.-B. Martens, and J. J. van Wijk, "A model of symbol size discrimination in scatterplots," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '10. New York, NY, USA: Association for Computing Machinery, 2010, pp. 2553–2562. [Online]. Available: <https://doi.org/10.1145/1753326.1753714>
- [25] J. Matejka, F. Anderson, and G. Fitzmaurice, "Dynamic opacity optimization for scatter plots," in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI '15. New York, NY, USA: Association for Computing Machinery, 2015, pp. 2707–2710. [Online]. Available: <https://doi.org/10.1145/2702123.2702585>
- [26] E. Bertini, L. Dell'Aquila, and G. Santucci, "Reducing InfoVis cluttering through non uniform sampling, displacement, and user perception," in *Visualization and Data Analysis 2006 – Proceedings of SPIE-IS and Electronic Imaging*, vol. 6060, 2006.
- [27] A. Mayorga and M. Gleicher, "Splatterplots: Overcoming overdraw in scatter plots," *IEEE Transactions on Visualization and Computer Graphics*, vol. 19, no. 9, pp. 1526–1538, Sep 2013.
- [28] M. C. Hao, U. Dayal, R. K. Sharma, D. Keim, and H. Janetzko, "Variable binned scatter plots," *Information Visualization*, vol. 9, no. 3, pp. 194–203, 2010.
- [29] F. V. Paulovich and R. Minghim, "HiPP: A novel hierarchical point placement strategy and its application to the exploration of document collections," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, pp. 1229–1236, Nov 2008.
- [30] Y. K. Leung and M. D. Apperley, "A review and taxonomy of distortion-oriented presentation techniques," *ACM Transactions on Computer-Human Interaction*, vol. 1, no. 2, pp. 126–160, Jun. 1994. [Online]. Available: <https://doi.org/10.1145/180171.180173>
- [31] M. Sarkar, S. S. Snibbe, O. J. Tversky, and S. P. Reiss, "Stretching the rubber sheet: A metaphor for viewing large layouts on small screens," in *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '93. New York, NY,

- USA: Association for Computing Machinery, 1993, pp. 81—91. [Online]. Available: <https://doi.org/10.1145/168642.168650>
- [32] D. A. Keim, M. C. Hao, U. Dayal, H. Janetzko, and P. Bak, "Generalized scatter plots," *Information Visualization*, vol. 9, no. 4, pp. 301–311, 2010. [Online]. Available: <https://doi.org/10.1057/ivs.2009.34>
- [33] D. A. Keim, C. Panse, M. Schäfer, M. Sips, and S. C. North, "HistoScale: An efficient approach for computing pseudo-cartograms," in *14th IEEE Visualization 2003 (VIS 2003)*. Piscataway, NJ: IEEE, 2003.
- [34] R. G. Raidou, M. E. Gröller, and M. Eisemann, "Relaxing dense scatter plots with pixel-based mappings," *IEEE Transactions on Visualization and Computer Graphics*, vol. 25, no. 6, pp. 2205–2216, June 2019.
- [35] J. O. Vollmer and J. Döllner, "2.5d dust & magnet visualization for large multivariate data," in *Proceedings of the 13th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '20. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: <https://doi.org/10.1145/3430036.3430045>
- [36] R. Cutura, C. Morariu, Z. Cheng, Y. Wang, D. Weiskopf, and M. Sedlmair, "Hagrid — gridify scatterplots with Hilbert and Gosper curves," in *Proceedings of the 14th International Symposium on Visual Information Communication and Interaction*, ser. VINCI '21. New York, NY, USA: Association for Computing Machinery, 2021. [Online]. Available: <https://doi.org/10.1145/3481549.3481569>
- [37] ———, "Hagrid: using Hilbert and Gosper curves to gridify scatterplots," *Journal of Visualization*, vol. 25, pp. 1291–1307, 2022.
- [38] G. M. Hilasaca, W. E. Marcilio-Jr, D. M. Eler, R. M. Martins, and F. V. Paulovich, "A grid-based method for removing overlaps of dimensionality reduction scatterplot layouts," *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–14, 2023.
- [39] X. Liu, Y. Hu, S. North, and H.-W. Shen, "CorrelatedMultiples: Spatially coherent small multiples with constrained multidimensional scaling," *Computer Graphics Forum*, vol. 37, no. 1, pp. 7–18, 2018. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12526>
- [40] G. W. Furnas, "Generalized fisheye views," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '86. New York, NY, USA: Association for Computing Machinery, Apr 1986, pp. 16–23.
- [41] C. Pindat, E. Pietriga, O. Chapuis, and C. Puech, "JellyLens: Content-aware adaptive lenses," in *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: Association for Computing Machinery, 2012, pp. 261–270. [Online]. Available: <https://doi.org/10.1145/2380116.2380150>
- [42] G. Ellis, E. Bertini, and A. Dix, "The sampling lens: Making sense of saturated visualisations," in *CHI'05 Extended Abstracts on Human Factors in Computing Systems, CHI EA'05*, 2005, pp. 1351–1354.
- [43] G. Ellis and A. Dix, "The plot, the clutter, the sampling and its lens: Occlusion measures for automatic clutter reduction," in *Proceedings of the Working Conference on Advanced Visual Interfaces*, ser. AVI '06. New York, NY, USA: Association for Computing Machinery, 2006, pp. 266–269. [Online]. Available: <https://doi.org/10.1145/1133265.1133318>
- [44] C. Tominski, S. Gladisch, U. Kister, R. Dachselt, and H. Schumann, "Interactive lenses for visualization: An extended survey," *Computer Graphics Forum*, vol. 36, no. 6, pp. 173–200, Sep 2017.
- [45] F. C. Crow, "Summed-area tables for texture mapping," in *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA: Association for Computing Machinery, 1984, pp. 207–212.
- [46] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2002.
- [47] S. Ehsan, A. F. Clark, N. ur Rehman, and K. D. McDonald-Maier, "Integral images: Efficient algorithms for their computation and storage in resource-constrained embedded vision systems," *Sensors (Basel)*, vol. 15, no. 7, pp. 16 804–16 830, Jul 2015.
- [48] N. Singhal, J. W. Yoo, H. Y. Choi, and I. K. Park, "Implementation and optimization of image processing algorithms on embedded GPU," *IEICE Transactions on Information and Systems*, vol. E95.D, no. 5, pp. 1475–1484, 2012.
- [49] G. Bradski, "The opencv library." *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, vol. 25, no. 11, pp. 120–123, 2000.
- [50] C. Reinbold and R. Westermann, "Parameterized splitting of summed volume tables," *Computer Graphics Forum*, vol. 40, no. 3, pp. 123–133, 2021.
- [51] R. Lienhart and J. Maydt, "An extended set of Haar-like features for rapid object detection," in *Proceedings. International Conference on Image Processing*, vol. 1, Sep 2002, pp. 900–903.
- [52] R. Lienhart, A. Kuranov, and V. Pisarevsky, "Empirical analysis of detection cascades of boosted classifiers for rapid object detection," in *Pattern Recognition*, B. Michaelis and G. Krell, Eds., vol. 2781. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 297–304.
- [53] A. L. C. Barczak, M. J. Johnson, and C. H. Messom, "Real-time computation of Haar-like features at generic angles for detection algorithms," *Research Letters in the Information and Mathematical Sciences*, vol. 9, pp. 98–111, 2006.
- [54] Tat-Jun Chin, Hanlin Goh, and Ngan-Meng Tan, "Exact integral images at generic angles for 2D barcode detection," in *2008 19th International Conference on Pattern Recognition*, Dec 2008, pp. 1–4.
- [55] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia, "3-dimensional pliable surfaces: For the effective presentation of visual information," in *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*, ser. UIST '95. New York, NY, USA: Association for Computing Machinery, 1995, pp. 217–226. [Online]. Available: <https://doi.org/10.1145/215585.215978>
- [56] S. Bachthaler and D. Weiskopf, "Continuous scatterplots," *IEEE Transactions on Visualization and Computer Graphics (Proceedings Visualization / Information Visualization 2008)*, vol. 14, no. 6, pp. 1428–1435, December 2008.
- [57] V. Molchanov, A. Fofonov, and L. Linsen, "Frequency-based progressive rendering of continuous scatterplots," *Journal of WSCG*, vol. 21, no. 1, pp. 49–58, July 2013.
- [58] C. D. Stolper, A. Perer, and D. Gotz, "Progressive visual analytics: User-driven visual exploration of in-progress analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 20, no. 12, pp. 1653–1662, 2014.
- [59] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [60] S. A. Nene, S. K. Nayar, and H. Murase, "Columbia Object Image Library (COIL-100)," Tech. Rep., February 1996. [Online]. Available: <http://www1.cs.columbia.edu/CAVE/software/softlib/coil-100.php>
- [61] J. Venna and S. Kaski, "Neighborhood preservation in nonlinear projection methods: An experimental study," in *Artificial Neural Networks—ICANN 2001*, G. Dorffner, H. Bischof, and K. Hornik, Eds. Berlin: Springer, 2001, pp. 485–491.
- [62] K. Misue, P. Eades, W. Lai, and K. Sugiyama, "Layout adjustment and the mental map," *Journal of Visual Languages & Computing*, vol. 6, no. 2, pp. 183–210, June 1995. [Online]. Available: <https://api.semanticscholar.org/CorpusID:611807>
- [63] D. Dua and C. Graff, "UCI Machine Learning Repository," 2019. [Online]. Available: <http://archive.ics.uci.edu/ml>
- [64] J. Cohen, *Statistical power analysis for the behavioral sciences*. Academic press, 2013.

Hennes Rave is a doctoral researcher in the Visualization and Graphics (VISIX) group at the University of Münster, Germany, where he received his Master's degree in Computer Science in 2021. His research interests include interactive visualization, spectral image visualization, and computer graphics.

Vladimir Molchanov is a post-doctoral researcher with University of Münster, Germany. He received his Bachelor's and Master's degrees in Applied Mathematics from Novosibirsk State University, Russia, and his Ph.D. degree in Mathematics from Jacobs University, Bremen, Germany. His research interests include projection methods, interactive exploratory systems, visualization of multidimensional data, and biomedical data analysis.

Lars Linsen is a Full Professor of Computer Science at the University of Münster, Germany. He received his academic degrees from the University of Karlsruhe, Germany, including a Ph.D. in Computer Science. Subsequent affiliations were the University of California, Davis, U.S.A., as a post-doctoral researcher and lecturer, the University of Greifswald, Germany, as an assistant professor, and Jacobs University, Bremen, Germany, as an associate and full professor. His research interests are in interactive visual data analysis.