Eurographics Conference on Visualization (EuroVis) 2025
W. Aigner, N. Andrienko, and B. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 44 (2025), Number 3

Sca^{2}Gri: Scalable Gridified Scatterplots

S. Frey

University of Groningen, Bernoulli Institute, The Netherlands

Abstract

Scatterplots are widely used in exploratory data analysis. Representing data points as glyphs is often crucial for in-depth investigation, but this can lead to significant overlap and visual clutter. Recent post-processing techniques address this issue, but their computational and/or visual scalability is generally limited to thousands of points and unable to effectively deal with large datasets in the order of millions. This paper introduces Sca^{2}Gri (Scalable Gridified Scatterplots), a grid-based post-processing method designed for analysis scenarios where the number of data points substantially exceeds the number of glyphs that can be reasonably displayed. Sca^{2}Gri enables interactive grid generation for large datasets, offering flexible user control of glyph size, maximum displacement for point to cell mapping, and scatterplot focus area. While Sca^{2}Gri's computational complexity scales cubically with the number of cells (which is practically bound to thousands for legible glyph sizes), its complexity is linear with respect to the number of data points, making it highly scalable beyond millions of points.

CCS Concepts
• ***Human-centered computing*** → ***Visualization;***

1. Introduction

Scatterplots are among the most commonly used visualizations in exploratory data analysis [SG18, YXX*21, MPOW17]. Glyphs representing individual data points such as images [ENP*09] or other complex visual encodings [CGSQ11] play a key role in detailed analysis but are subject to substantial overlap and visual clutter. As the overlap between glyphs representing data instances increases, scatterplots become less effective [SMT13, SG18, YXX*21]. This overlap impairs comprehension [MG13a] and diminishes the accuracy of analytical tasks [Elm05]. The issue is exacerbated when glyphs carry more detailed information.

Approaches to address this problem can be divided into two categories: (1) overlap-free and (2) overlap-removal strategies. While overlap-free techniques aim to generate layouts without overlap [PdOdAL09, Pd09], overlap-removal methods focus on rearranging glyphs in a post-processing step to eliminate overlaps in a given scatterplot while preserving the original layout characteristics as much as possible. A major conceptual advantage of overlap-removal approaches is that they are agnostic to how the scatterplots are generated. For example, they can be combined with common dimensionality reduction (DR) approaches like UMAP [MHM20] which are widely used to uncover patterns in high-dimensional datasets.

Prior works on overlap-removal strategies employed cost-function optimization [GRP*14, GH10], triangulation [NNB*16], orthogonal scan-line algorithms [MELS95, HIMF98, DMS06, GPNB17], and grid-based methods [CMC*22, HMJE*24]. However, they lack the capabilities to computationally and/or visually deal with a large number of points—hundreds of thousands, millions, and more—that exceed what can adequately be presented directly with a grid.

This work proposes Sca^{2}Gri (Scalable Gridified Scatterplots), an overlap-removal method rapidly assigning millions of points $\mathcal{P}$ to grid cells $\mathcal{C}$. Addressing both visual and computational limitations, Sca^{2}Gri focuses on the efficient handling and interactive exploration of scenarios where $\|\mathcal{P}\| >> \|\mathcal{C}\|$. This is highly relevant for large data analysis as the feasible number of grid cells $\|\mathcal{C}\|$ is generally limited to several hundreds at most—otherwise there are too many and too small cells for effective glyph display—while $\|\mathcal{P}\|$ can easily reach thousands to millions and beyond. You can find Sca^{2}Gri's code at `https://github.com/freysn/sca2gri`. We consider the main contributions of Sca^{2}Gri to be as follows:

Scalability to Large Data. Fast ($\lessapprox 50$ ms) generation of grids with millions of points $\|\mathcal{P}\|$, targeting large data scenarios where $\|\mathcal{P}\| >> \|\mathcal{C}\|$. For this, Sca^{2}Gri's novel scheme crucially exhibits linear complexity regarding $\|\mathcal{P}\|$.

Strict Upper Bound on Displacement. Threshold τ restricts the maximum displacement in the assignments of points to cells.

Flexible Interactive Control. Users can interactively control the generation of grids with millions of points via three intuitive parameters: scatterplot area (z), (horizontal) grid resolution (g_x), and displacement relative to the presented area (τ_z).

High-Quality Grids. Generation of high-quality gridified scatterplots with gaps, demonstrating competitive performance to SOTA for traditional $\|\mathcal{P}\| \approx \|\mathcal{C}\|$ scenarios across various metrics.

2. Related Work

Occlusion in scatterplots is a widely recognized problem that negatively affects the clarity of the visualization [Bra14, SMT13, SG18, YXX*21], impacts comprehension [MG13a], and reduces accuracy in analytical tasks [Elm05]. Various strategies have been proposed to mitigate this issue. In an early work, Carr et al. [DBCL87] highlight the importance of dealing with overplotting in large scatterplots and discuss generally applicable interactive techniques like subset selection and animation. Another direction aims to adapt the glyph which represent individual points. Here, common approaches include adjusting transparency [MPOW17], resizing [LvM09, LMvW10], sampling [CGZ*20, BS06], or employing density and contour plots [CLNL87, TSW*07, MG13b]. However, the two most common strategies for resolving occlusion are so-called overlap-free and overlap-removal approaches: overlap-free techniques directly create layouts without overlap, while overlap-removal methods rearrange glyphs in a post-processing step while maintaining the original layout characteristics to the largest degree possible. Sca^{2}Gri belongs to the class of overlap-removal methods.

Early overlap-free layouts include IncBoard and HexBoard [PdOdAL09, Pd09]. Meulemans et al. [MDS*17] explore the design space offered by small multiples with gaps by means of various metrics, and in a later work propose a three-step pipeline for generating coherent grid maps [MSS21]. Li et al. [LSL*23] propose an overlap-free method based on a dual space coupling model which integrates a geometry-based data transformation algorithm, an efficient spatial mutual exclusion guided view transformation algorithm, and an overlap-free oriented visual encoding configuration model and a radius adjustment tool. Various distance-preserving grid layouts have been proposed that do not use scatterplot layouts as inputs [AR88, SG11, vKS04, Rai34, MH17, WDS10, EvSS13], but we consider them to be outside of the scope of this work. That being said, some of these techniques, such as Eppstein's point-matching method [EvSS13], minimize displacement between original and grid-aligned points, similar to how Sca^{2}Gri minimizes distortions.

Overlap-removal strategies like IsoMatch [FDH*15], Kernelized Sorting [QSST10], and NMAP [DSF*14] use grid-based approaches to map scatterplot points into orthogonal cells while preserving distance. RWordle [SSS*12] utilizes bounding boxes and a spiral search for empty spaces. ProjSnippet [GRP*14] maximizes an energy function to maintain similarity relationships while removing overlaps. Other techniques, such as MIOLA [GCNT13], PRISM [GH10], and GTree [NNB*16] employ optimization methods or triangulation, although they potentially introduce significant distortions, reduce glyph size, or suffer from instability in dense layouts. Hashedcubes enables the real-time generation of binned scatterplots, linked histograms and heatmaps for the visual exploration of large datasets [PSSC17].

Rearranging layouts by axis is another common approach. For example, PFS [MELS95, HIMF98] moves glyphs horizontally and vertically to eliminate overlaps, but does not preserve the aspect ratio. VPSC [DMS06] defines non-overlap constraints for each axis, but can significantly distort dense scatterplots. Recently, Hagrid [CMC*22] uses space-filling curves to assign points to grid cells, while ReArrange [GPNB17] employs a line-sweep algorithm to address overlaps with minimal displacement, albeit at the cost of distorting the aspect ratio. [BHJS23] propose a metric to evaluate the quality of an arrangement based on user experiments, and introduce a new algorithm for creating visually sorted grid layouts.

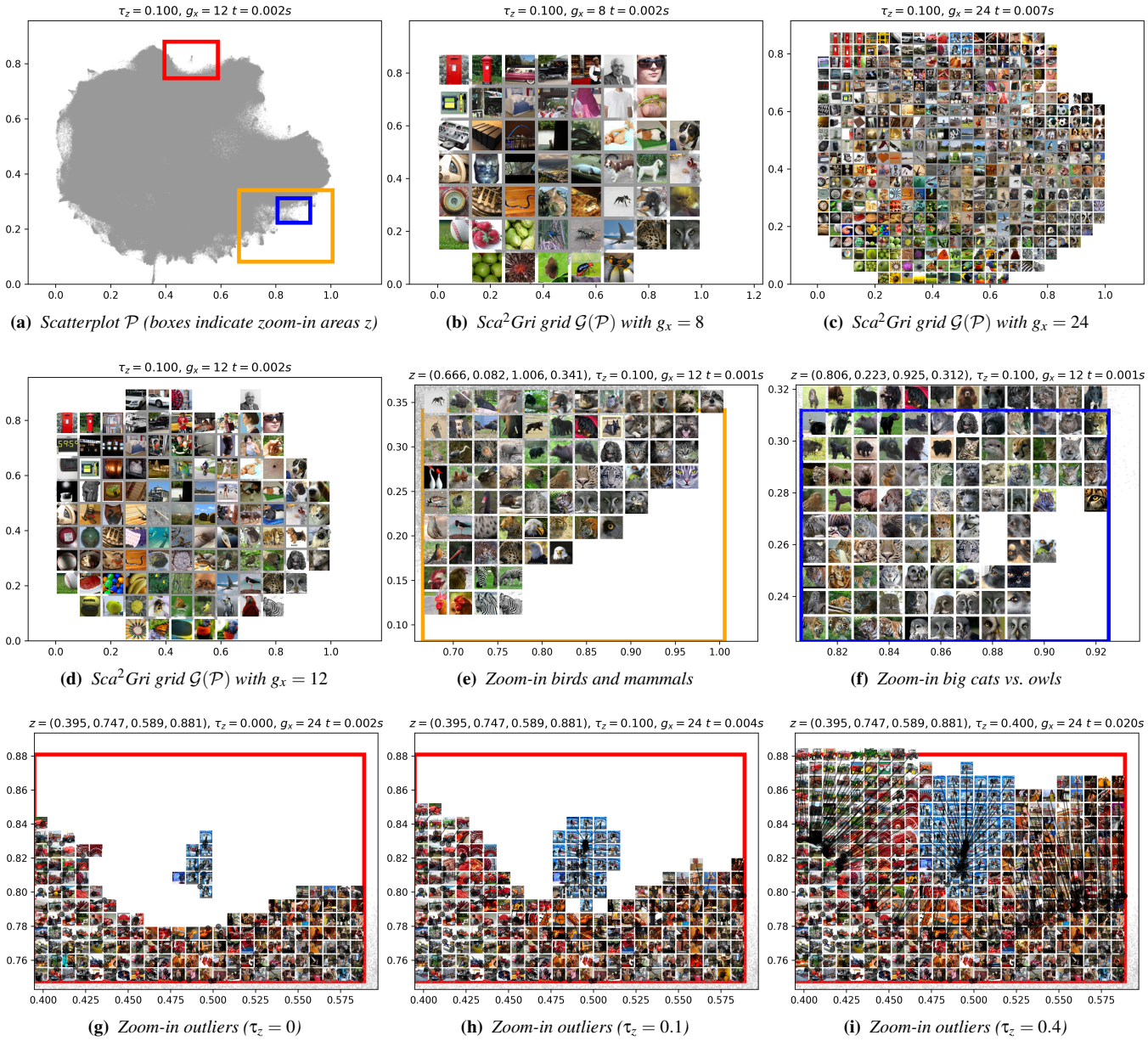
Several grid-based methods focus on similarity preservation. For instance, Chen et al. [CYL*21] use the Jonker-Volgenant algorithm to assign projected samples to grid cells while minimizing distant connections (we also employ Jonker-Volgenant in our implementation of Sca^{2}Gri). Cluster-aware grid layouts [ZYC*24] aim to maintain cluster structures by optimizing proximity and compactness. Techniques like Self-Organizing Maps (SOMs) [Koh90] and Kernelized Sorting [QSST10] assign members to grid cells based on similarity, but hash collisions and overlaps remain a challenge. Hilasaca et al. [HMJE*24] propose a post-processing strategy (DGrid) to remove overlaps while aiming to faithfully preserve the original layout's characteristics and bounding the minimum glyph sizes. They demonstrate that DGrid surpasses the state-of-the-art in overlap removal through an extensive comparative evaluation involving various metrics. We employ a similar approach in our evaluation and compare Sca^{2}Gri against DGrid and Hagrid below. NMAP [DSF*14] and DGrid [HMJE*24] use space partitioning and multidimensional projections, while IsoMatch [FDH*15] employs a bipartite graph and the Hungarian method for grid assignments. However, these techniques can be computationally expensive and do not scale well to large numbers of points.

In general, grid layouts typically face scalability issues, particularly when the number of data points substantially exceeds the number of grid cells that can effectively be shown. Hierarchical techniques have been introduced to address this, organizing large datasets into manageable grids while maintaining neighborhood relationships and grid uniformity [Fre22]. Extensions beyond 2D grids, such as hierarchical clustering [Joh67] or treemaps [WD08], offer scalable solutions but cannot preserve the scatterplot structure.

In contrast to other grid layout techniques, Sca^{2}Gri specifically targets scenarios with a significantly larger number of points $\|\mathcal{P}\| := \|\mathcal{P}\|$ in relation to the number of grid cells $\|\mathcal{C}\| := \|\mathcal{C}\|$, i.e. $\|\mathcal{P}\| >> \|\mathcal{C}\|$. It controls grid size, aspect ratio, and displacement, producing overlap-free layouts with minimal distortions and preserving distance and neighborhood relationships. Unlike approaches that inefficiently use visual space or distort layouts, Sca^{2}Gri ensures that visualizations remain readable and scalable. To the best of our knowledge, Sca^{2}Gri is the first approach to generate overlap-free layouts for scatterplots beyond millions of points at interactive rates.

3. Objective

Sca^{2}Gri aims to present overlap-free glyphs for scatterplots with a large number of points $\mathcal{P}$. It considers a grid with (non-overlapping) cells $\mathcal{C}$, whereas each grid cell contains a glyph or image representing a point. We restrict ourselves to uniform grids in this work, the extension of Sca^{2}Gri to arbitrary grids is briefly discussed in Sec. 9. The goal is to determine an assignment $\mathcal{G} : \mathcal{P} \to \mathcal{C}$ that minimizes the displacement when "moving" points $p \in \mathcal{P}$ to grid cells $c \in \mathcal{C}$ (considering the Euclidean distance to the cell center). The allowed displacement should be constrained by threshold τ, with the exception that a point p may always be assigned to the grid cell c that it is located in. This is denoted as $p \in \chi[c]$ in the following, with $\chi : \mathcal{C} \to \mathcal{P}$ mapping cells to the points that they contain.



(a) *Scatterplot $\mathcal{P}$ (boxes indicate zoom-in areas z)* **(b)** *Sca^{2}Gri grid $\mathcal{G}(\mathcal{P})$ with $g_x = 8$* **(c)** *Sca^{2}Gri grid $\mathcal{G}(\mathcal{P})$ with $g_x = 24$*

(d) *Sca^{2}Gri grid $\mathcal{G}(\mathcal{P})$ with $g_x = 12$* **(e)** *Zoom-in birds and mammals* **(f)** *Zoom-in big cats vs. owls*

(g) *Zoom-in outliers ($\tau_z = 0$)* **(h)** *Zoom-in outliers ($\tau_z = 0.1$)* **(i)** *Zoom-in outliers ($\tau_z = 0.4$)*

Figure 1: *Sca^{2}Gri exploration example with the 1.2 Million images from the Imagenet dataset (please zoom in to enlarge). Starting with the scatterplot from UMAP projection (a), it demonstrates interactive adjustment of the three Sca^{2}Gri parameters scatterplot* grid resolution *g_x (b–d),* scatterplot area *z (e–g), and* displacement *τ_z (g–i). The colored boxes in (a) correspond to zoom-ins during the exploration from (e) to (i). Gray points depict $\mathcal{P}$, the images are shown in the cells $\mathcal{C}$ (size is scaled to 90% to leave boundary space), and black semitransparent lines indicate displacement when mapping scatterplot points to cells in g–i. Parameters and timings are listed at the top of each frame.*

With this, Sca^{2}Gri needs to solve the following optimization problem, minimizing the sum of displacements when mapping points to cells via assignment $\mathcal{G}$:

$$\begin{aligned} \min_{\mathcal{G}} \quad & \sum_{p \in \mathcal{P}} \|p - \mathcal{G}[p]\| \\ \text{s.t.} \quad & (\|p - \mathcal{G}[p]\| \leq \tau) \vee (p \in \chi[\mathcal{G}[p]]) \end{aligned} \tag{1}$$

Crucially, Sca^{2}Gri targets scenarios where the number of points $\|\mathcal{P}\|$ significantly exceeds the number of cells $\|\mathcal{C}\|$ that can expressively be presented with a grid (i.e., $\|\mathcal{P}\| >> \|\mathcal{C}\|$). This means that generally only a subset of points $\subseteq \mathcal{P}$ can be assigned to grid cells $\mathcal{C}$, but also vice versa not all grid cells potentially have an associated point due to the displacement threshold (and can be considered to be empty from a presentation point of view).

To allow flexible (interactive) control, Sca^{2}Gri exhibits three

Sca²Gri Step	Short Description	Input	Output	Complexity
***reduce**-grid*	determine list / number of points contained by each grid cell	$\mathcal{P}, \mathcal{C}$	$\chi : \mathcal{C} \to \mathscr{P}(\mathcal{P})$	$O(\|\mathcal{P}\|)$
***reduce**-onion*	compute layers of equidistant neighbor cells	$\mathcal{C}, \tau$	neighborhood onion $\mathcal{N}$	$O(\|\mathcal{C}\| \log \|\mathcal{C}\|)$
***reduce**-bound*	identify points to keep per cell	$\chi, \mathcal{N}$	reduced points $\mathcal{P}^*$	$O(\|\mathcal{C}\|^2)$
***assign**-cost*	compute # assignment elements e and cost matrix $\mathcal{M} : e \times e$	$\mathcal{P}^*, \mathcal{C}, \tau$	cost matrix $\mathcal{M}$	$O(\|\mathcal{C}\|^2)$
***assign**-LA*	solve linear assignment problem (Jonker-Volgenant)	$\mathcal{M}$	assignment $\mathcal{G} : \mathcal{P}^* \to \mathcal{C}$	$O(\|\mathcal{C}\|^3)$
($\|\mathcal{P}\|$ depicts the number of points, $\|\mathcal{C}\|$ denotes the number of grid cells)				$O(\|\mathcal{P}\| + \|\mathcal{C}\|^3)$

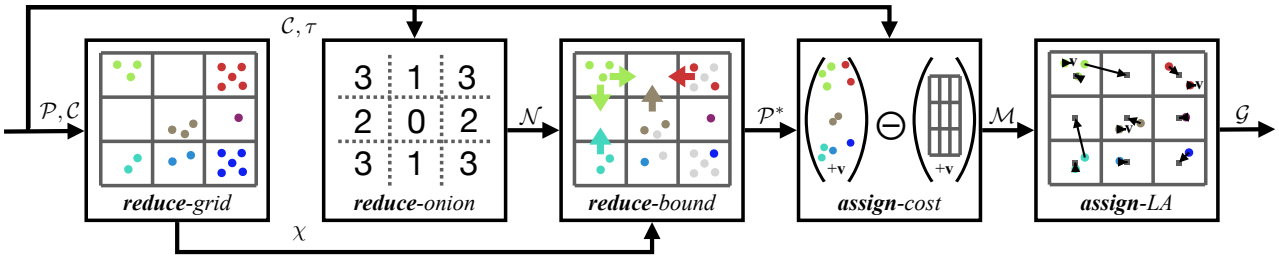


Figure 2: *Sca²Gri steps to determine the assignment $\mathcal{G}$ of points $\mathcal{P}$ to grid cells $\mathcal{C}$ (see table (top) for step characteristics and figure (bottom) for an illustration of the process). The grid $\mathcal{C}$ is created under consideration of scatterplot area z and cell size g_x. **reduce**-grid identifies for each point $p \in \mathcal{P}$ in which cell c of the grid C it is located (color indicates cell correspondence in the illustration). **reduce**-onion creates neighborhood onion $\mathcal{N}$ by grouping the cell neighborhood into layers which are equidistant to the center of the origin cell and sorted in the order of increasing distance. The maximum distance of cells that $\mathcal{N}$ contains is bound by displacement threshold τ. As we consider uniform grids in this work, $\mathcal{N}$ only needs to be created once regardless of how many cells there are in the grid. The example above features non-square cells (with $s_x > s_y$) and numbers 0 to 3 indicate different layers, with 0 being the respective cell itself, 1 its closest neighbors, etc. In **reduce**-bound, cells iterate over $\mathcal{N}$ to determine an upper bound on how many of its associated points can be assigned to grid cells $\mathcal{C}$. A reduced set of points $\mathcal{P}^* \subseteq \mathcal{P}$ is created accordingly. **assign**-cost stores point→cell assignment cost in a quadratic cost matrix $\mathcal{M}$ with $e \times e$ elements on either side (i.e,. e points and e cells), introducing **v**irtual points and/or cells as needed for balancing (indicated via **v** in the illustration). Finally, **assign**-LA solves the linear assignment problem with $\mathcal{M}$ to yield the assignment of points to grid cells $\mathcal{G}$.*

parameters to adapt the presentation of the grid: scatterplot area z, horizontal grid resolution g_x, and (relative) displacement τ_z (please find an example demonstrating parameter changes in Fig. 1):

Scatterplot Area $\mathbf{z}$ $:= \left(\mathbf{z}_\mathbf{x}^{\text{from}}, \mathbf{z}_\mathbf{y}^{\text{from}}, \mathbf{z}_\mathbf{x}^{\text{to}}, \mathbf{z}_\mathbf{y}^{\text{to}}\right)$ determines the region of interest. As not all points $\mathcal{P}$ can be shown in a single grid view in cells $\mathcal{C}$, it is crucial to allow users to select different areas of a scatterplot to explore. For example, starting from an overview, different parts can be explored in detail by zooming in (e.g., by dragging the mouse to select a rectangular area).

Grid Resolution $\mathbf{g_x}$ specifies the horizontal resolution of the grid in the focus area z. With this, the horizontal cell size s_x is accordingly determined by $s_x := (z_x^{\text{to}} - z_x^{\text{from}})/g_x$. The cell size in y direction s_y is then simply computed with the aspect ratio of the glyph or image that is to be shown: $s_y := s_x \cdot [\text{glyph height}]/[\text{glyph width}]$. Finally, the vertical grid resolution g_y is determined via $g_y := \lceil z_y / s_y \rceil$.

Displacement $\tau_\mathbf{z}$ determines the maximum distance of a point to a cell (center) with which assignment is still possible. Decoupling grid resolution g_x and displacement bound τ_z allows for flexible tuning of representations, with τ_z essentially controlling a trade-off: larger τ_z means more data points can be shown (as there are generally fewer empty cells), while lower τ_z yields a closer preservation of the scatterplot structure. To be more intuitive, τ_z is provided in relation to scatterplot area z such that the setting qualitatively speaking is agnostic to changes in z. The absolute displacement value τ – as used in Eq. 1 – is simply computed from τ_z by multiplying with the horizontal extent of the scatterplot area: $\tau := \tau_z \cdot (z_x^{\text{to}} - z_x^{\text{from}})$.

For the exploration of large datasets, it is crucial that all these parameters can be interactively controlled. Fig. 1 provides an example of such an analysis session, it will be discussed in more detail later in Sec. 6. To achieve the required interactive performance with large scatterplots, Sca²Gri's computational scaling with respect to the number of points $\|\mathcal{P}\|$ needs to be linear. The method to achieve this will now be described in detail in the upcoming Sec. 4.

4. Method

Sca²Gri follows a two-phase approach to exploit the structure of the problem and achieve the objectives as outlined in Sec. 3:

reduce*.** Decrease the number of points from $\mathcal{P}$ to $\mathcal{P}^*$ under consideration of all three parameters. The main idea is to only keep as many points as could theoretically be presented in the grid (i.e., assigned to grid cells during the ***assign phase). Crucially, we aim to achieve $O(\|\mathcal{P}^*\|) = O(\|\mathcal{C}\|)$ while only inducing negligible impact on the final quality of the produced assignment (bound by the cell size s for each point-cell pair).

***assign*.** Map the (reduced) set of points $\mathcal{P}^*$ to the cells of the grid $\mathcal{C}$ under consideration of the maximum bound for displacement τ, effectively solving Eq. 1.

The procedure will now be discussed in detail below in Sec. 4.1

(***reduce***) and Sec. 4.2 (***assign***). Please find an overview of individual steps and their characteristics in Fig. 2. An analysis of the resulting total complexity of Sca²Gri is provided in Sec. 4.3.

4.1. *reduce*

This phase determines the reduced set of points $\mathcal{P}^* \subseteq \mathcal{P}$ which is further considered in ***assign***. Note that ***reduce*** can only have an effect when not all points can be assigned to cells, i.e., when $\|\mathcal{P}\| > \|\mathcal{C}\|$ and/or $\tau \neq \infty$. If this criterion is not met, Sca²Gri simply skips this phase and runs ***assign*** with $\mathcal{P}^* = \mathcal{P}$. ***reduce*** consists of three individual steps that will now be discussed in detail in the following: ***reduce***-*grid*, ***reduce***-*onion*, and ***reduce***-*bound*.

reduce-grid

First, we identify for each point $p \in \mathcal{P}$ whether it lies within zoom area z and, if so, in which cell c of the grid C it is located. As we restrict ourselves to uniform grids in this work, this is simply determined by offsetting p with the minimum domain bounds from z, dividing by cell size s, and taking the largest integer smaller than the result (the `floor` operation) in both x and y direction: $\left(\lfloor (p - z_x^{\text{from}}) \cdot s_x \rfloor, \lfloor (p - z_y^{\text{from}}) \cdot s_y \rfloor\right)$. We compile for each cell the list of points $\mathcal{P}_c \subset \mathcal{P}$ that it contains: $\chi : \mathcal{C} \to \mathcal{P}_c$. Note that in our implementation, we only keep a maximum of $\|\chi[c]\| = \|\mathcal{C}\|$ for each cell c (it is not possible to eventually assign more points than there are grid cells $\|\mathcal{C}\|$ in total).

Computational Complexity. For each individual point, its corresponding grid cell can be found in constant time. Accordingly, the computational complexity of this step is $O(\|\mathcal{P}\|)$.

reduce-onion

Neighborhood onion $\mathcal{N}$ consists of different layers that contain relative indices to all neighbors with equal distance, whereas layers $L \in \mathcal{N}$ are sorted in the order of increasing distance. For this, it considers the width-to-height ratio s_x/s_y of a cell, e.g., $\mathcal{N} = \{\{(0,0)\},\{(1,0),(0,-1)\},\{(0,1),(-1,0)\},\dots\}$ for cells with $s_x > s_y$. Trivially, it can be computed by sorting all possible neighbors with respect to distance. Neighborhood onions contain layers up to a maximum distance of $\tau + \frac{\|s\|}{2}$.

Computational Complexity. As we consider a uniform grid in this work, the onion only needs to be determined once for the whole grid. Accordingly, the computational complexity is $O(\|\mathcal{C}\| \log \|\mathcal{C}\|)$.

reduce-bound

The core idea of this step is that for each set of points $\chi[c]$ associated with a cell c we determine a theoretical upper bound $\mu[c]$ regarding how many of these points can maximally be assigned to grid cells $\mathcal{C}$. This allows to substantially reduce the set of points from $\mathcal{P}$ to $\mathcal{P}^* \subseteq \mathcal{P}$ such that $\|\mathcal{P}^*\| << \|\mathcal{P}\|$ when the number of cells is significantly smaller than the number of points: $\|\mathcal{C}\| << \|\mathcal{P}\|$.

In our method to determine $\mathcal{P}^*$ (Alg. 1), we loop over the layers $L \in \mathcal{N}$ of the neighborhood onion from ***reduce***-*onion* (line 4,line 5) and all cells in our current working set C' (line 6). C' initially contains all cells of the grid (line 1) and through the course of the algorithm we keep track of which cells still need to be considered in the subsequent iteration.

Input: points per cell χ (***reduce***-*grid*)
Input: neighborhood onion $\mathcal{N}$ (***reduce***-*onion*)
Result: reduced set of points $\mathcal{P}^*$
1 $\mathcal{C}' \leftarrow \mathcal{C}$; // working set of considered cells
2 $\sigma[\mathcal{C}] \leftarrow \infty$; // layer index indicating when a cell is visited
3 $\mu[\mathcal{C}] \leftarrow 0$; // number of points to keep per cell
4 **forall** $l \in 1 \dots \|\mathcal{N}\|$ **do**
5 $L \leftarrow \mathcal{N}[l]$;
6 **forall** $c \in \mathcal{C}'$ **do**
7 $T_l^c \leftarrow \{c_\Delta := (c+\Delta)\ \forall \Delta \in L \mid c_\Delta \in \mathcal{C}, l \leq \sigma[c_\Delta]\}$
8 $r \leftarrow \mu[c] + \|T_l^c\|$ // # points required from c
9 **if** $\|\chi[c]\| \geq r$ **then**
10 $\sigma[T_l^c] \leftarrow l$; // mark visited cells with layer index
11 $\mu[c] \leftarrow r$; // r is new upper bound for c
12 **else**
13 $\mu[c] \leftarrow \|\chi[c]\|$; // c keeps all associated points
14 **if** $\|\chi[c]\| \leq r$ **then**
15 $\mathcal{C}' \leftarrow \mathcal{C}' - \{c\}$; // omit cell in subsequent layers
// output combined subset of $\mu[c]$ points across cells $c \in \mathcal{C}$
16 **return** $\mathcal{P}^* := \cup_{c \in \mathcal{C}} (P_c^* \text{ with } P_c^* \subseteq \chi[c], \|P_c^*\| = \mu[c])$;

Algorithm 1: ***reduce***-*bound*. Determine the reduced set of points $\mathcal{P}^*$ based on the number of points $\chi[c]$ that can maximally be assigned to the grid across all cells $c \in \mathcal{C}$.

For each $c \in C'$, the set of corresponding cells in the current layer $T_l^c \subseteq \mathcal{C}$ is determined via $(c_\Delta := c + \Delta)\ \forall \Delta \in L$) and two constraints. First, the cells should be part of the grid ($c_\Delta \in \mathcal{C}$). Second, the cells must not have been visited in an earlier layer ($l \leq \sigma[c_\Delta]$)), where l depicts the index of the layer and σ marks the index when a cell has been visited first (initialized to infinity in line 2). Note that a cell c_Δ can be considered by multiple cells c in the same layer iteration l.

$\|T_l^c\|$ in combination with the number of points accounted for from prior layers $\mu[c]$ determines the total number of points r from c that could potentially be assigned to cells up until the current layer (line 8). If the number of points $\chi_n(c)$ associated with cell c is sufficient to cover r (line 9), we mark the cells T_l^c as visited with layer index l (line 10) and increase the upper bound $\mu[c]$ accordingly (line 11). Otherwise, the cell keeps all associated points (line 13). For the sake of efficiency, if all associated points are now accounted for after processing this layer (line 14), the cell c is removed from the working set for subsequent iterations (line 15).

Finally, after completing all layers, we compile the set of reduced points $\mathcal{P}^*$ by randomly choosing a subset of size $\mu[c]$ (the upper bound) from $\chi[c]$ (the points associated with c) for each cell c (line 16).

Computational Complexity. The complexity is $O(\|\mathcal{C}\|^2)$ as each grid cell potentially visits each other grid cell. We can further determine an upper bound of the total number of reduced points $\|\mathcal{P}^*\| = \sum_{c \in \mathcal{C}} \mu[c]$ as follows. In the uniform grid that we consider, there are maximally four cells that can have the same distance to any given cell $c \in \mathcal{C}$. This means that each cell can maximally be encountered by four other cells with the same layer index l. Accordingly, $\|\mathcal{P}^*\| := \|\mathcal{P}^*\| \leq 4\|\mathcal{C}\|$, and with this $O(\|\mathcal{P}^*\|) = O(\|\mathcal{C}\|)$.

4.2. *assign*

We now solve the optimization problem from Eq. 1, assigning the reduced point set $\mathcal{P}^*$ to grid cells $\mathcal{C}$. This is structured into two individual steps: ***assign**-cost* and ***assign**-LA*.

assign-cost

To map points to cells in Sca²Gri, we set up a balanced assignment problem with e elements on either side (points or cells), introducing virtual points or cells as needed. e is determined such that it is as small as possible (to minimize computational cost in the final step ***assign**-LA*), while still allowing to meet displacement constraint τ:

$$e = \begin{cases} \max(\|\mathcal{P}^*\|, \|\mathcal{C}\|), & \tau = \infty \\ \|\mathcal{P}^*\| + \|\mathcal{C}\| - n_{\boxtimes}, & \tau \neq \infty \end{cases} \tag{2}$$

In the case of $\tau = \infty$ (unlimited displacement), e is simply the maximum of the number of reduced points $\|\mathcal{P}^*\|$ and cells $\|\mathcal{C}\|$, whereas the side with fewer elements is filled with virtual elements to balance the problem. This means that with $\|\mathcal{P}^*\| > \|\mathcal{C}\|$ (more points than cells), all cells will be assigned to points ($\|\mathcal{P}\| = e$), and the rest to virtual points. Vice versa, when $\|\mathcal{C}\| > \|\mathcal{P}^*\|$ (more cells than points), all points are assigned to cells, and $\|\mathcal{P}\| = e$.

$\tau \neq \infty$ induces an assignment constraint which implies that both points and cells potentially need to be assigned to virtual elements (not all points can be matched with all cells and vice versa). On the extreme, if it would be possible that no point-cell pairs can meet this constraint, it would follow that $e = \|\mathcal{P}^*\| + \|\mathcal{C}\|$ (i.e., $\|\mathcal{P}^*\|$ virtual cells and $\|\mathcal{C}\|$ virtual points would need to be added). However, according to Eq. 1, in Sca²Gri it should always be possible to map a point p to its associated cell $\chi[p]$, regardless of τ. This allows to reduce e by the number of cells associated with at least one point $n_{\boxtimes} (:= \sum_{c \in \mathcal{C}} (\|\chi[c]\| > 0))$, utilizing that there will be at least as many valid assignment pairs for which no virtual elements are required. This means that eventually $e - \|\mathcal{C}\|$ virtual cells and $e - \|\mathcal{P}^*\|$ virtual points are added.

Finally, we compute the $e \times e$ cost matrix $\mathcal{M}$:

$$\mathcal{M}(p,c) = \begin{cases} \tau + \epsilon, & p \text{ or } c \text{ are virtual} \\ \min(\|p-c\|, \tau), & p \in \chi[c] \\ \|p-c\| & \text{else} \end{cases} \tag{3}$$

As depicted in Eq. 1, the cost of assigning a point $p \in \mathcal{P}^*$ to a cell in $c \in \mathcal{C}$ is their Euclidean distance (with respect to the center of a cell c). To reflect that each cell that contains a point should also represent one, we set the assignment cost to $\min(\tau - \epsilon, \|p-c\|)$ for the cell c a point p is associated with. The cost when either a virtual point or cell is involved is set to τ. Note that when $\tau = \infty$, a finite value larger than the maximum distance between any non-virtual point-cell pairs is used in the implementation.

Computational Complexity. Only distances between real (non-virtual) cells and points need to be computed. With this, the computational complexity for determining the cost matrix $\mathcal{M}$ is $O(\|\mathcal{P}^*\| \cdot \|\mathcal{C}\| = O(\|\mathcal{C}\|^2)$ (with $O(\|\mathcal{P}^*\|) = O(\|\mathcal{C}\|)$, see above).

assign-LA

Sca²Gri employs a modified Jonker-Volgenant algorithm with no initialization [Cro16]. It optimally solves the linear assignment problem to map $\mathcal{P}^*$ to $\mathcal{C}$ with the $e \times e$ cost matrix computed during ***assign**-cost* to yield the assignment of points to grid cells $\mathcal{G}$.

Computational Complexity. The Jonker-Volgenant has cubic complexity regarding the number of elements (i.e., $O(e^3)$ in our case). As $\|\mathcal{P}^*\|$ is in $O(\|\mathcal{C}\|)$ (see Sec. 4.1), also e is in $O(\|\mathcal{C}\|)$. Accordingly, this step exhibits an overall complexity of $O(\|\mathcal{C}\|^3)$.

4.3. Computational Complexity

When considering all individual steps, the total complexity yields

$$\begin{aligned} O(&\|\mathcal{P}\|[\textbf{\textit{reduce}}\textit{-grid}] && + \|\mathcal{C}\| \log \|\mathcal{C}\|[\textbf{\textit{reduce}}\textit{-onion}] \\ &+ \|\mathcal{C}\|^2[\textbf{\textit{reduce}}\textit{-bound}] && + \|\mathcal{P}\|[\textbf{\textit{reduce}}\textit{-select}] \\ &+ \|\mathcal{C}\|^2[\textbf{\textit{assign}}\textit{-cost}] && + \|\mathcal{C}\|^3[\textbf{\textit{assign}}\textit{-LA}]) \\ &= \underline{O(\|\mathcal{P}\| + \|\mathcal{C}\|^3).} \end{aligned} \tag{4}$$

This means that the computational complexity of Sca²Gri is linear with respect to the number of points $\|\mathcal{P}\|$ in the scatterplot, and cubic regarding the number of cells $\|\mathcal{C}\|$. Linear scalability with $\|\mathcal{P}\|$ is crucial for large datasets including many thousands, millions, or even more members, while $\|\mathcal{C}\|$ is strictly limited to several hundreds at most in practice—the glyphs presented within the cells otherwise become too numerous and too small. Note that when omitting ***reduce*** and with this $\mathcal{P}^* = \mathcal{P}$, the total complexity would be $O(\|\mathcal{P}\|^3 + \|\mathcal{C}\|^3)$, i.e., also cubic in the in the number of points — as all points need to be considered in ***assign**-LA* — which quickly becomes prohibitive for large sets of points (> thousands).

5. Evaluation Setup

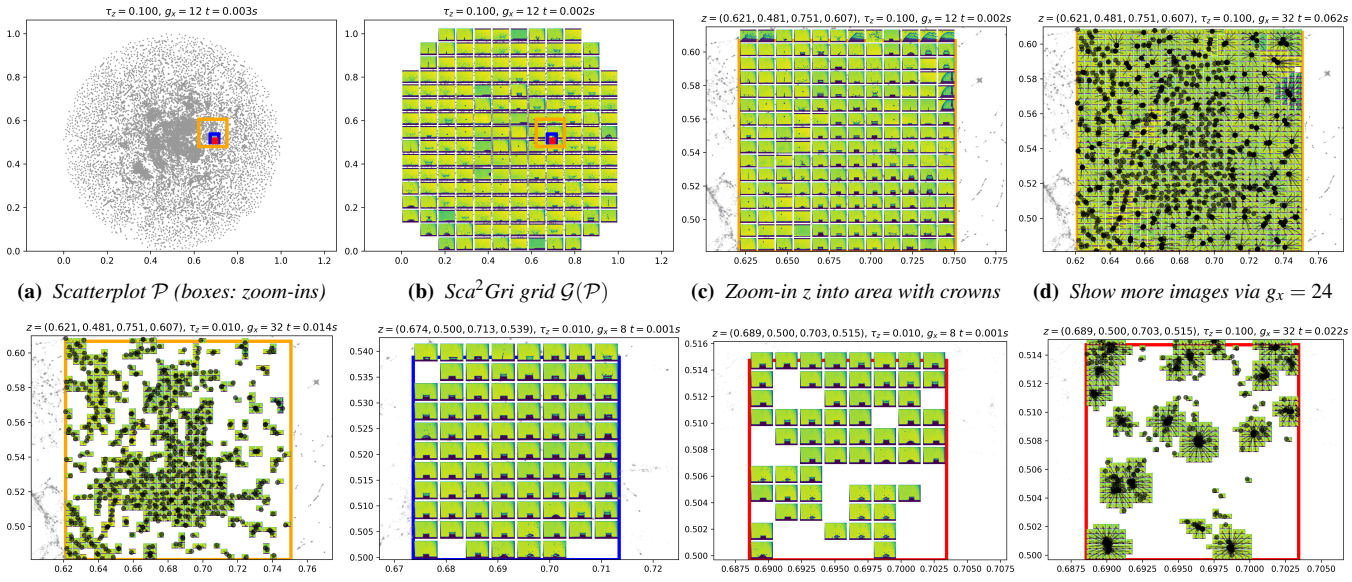
The implementation of Sca²Gri used in the following evaluation employs Python3.12 and is heavily based on NumPy [HMvdW*20], SciPy [VGO*20], and Numba [LPS15]. All measurements reported throughout this paper stem from running Sca²Gri on a single CPU core of Mac Mini M4 and 16 GB memory. The only exception is ***reduce**-grid* which is implemented using Apple Metal and runs in parallel on the M4 GPU.

Both synthetic and real-world data is considered in our evaluation. Without restricting generality, we normalize the scatterplots such that the spatial extent in x direction is in the $[0,1]$ range.

Real-World Data

ImageNet. We consider the ImageNet-1K dataset, primarily used for image classification tasks in computer vision. It has been extensively used for benchmarking image classification models and had a critical role in the development of deep learning architectures like AlexNet [KSH17], VGG [LD15], ResNet [HZRS16], and others. In our evaluation, we consider the 1.2 million training images resized to 64×64, with feature vectors extracted via the pre-trained ResNet18 model. These features are then projected using UMAP [MHM20].

Droplet experiments. This dataset contains an experimental parameter study that was conducted to analyze the impact of a droplet hitting a thin fluid film [GCM*16]. It consists of a total of 863 610 experimental images at a resolution of 430 px × 320 px. Here, we first extract perceptual hashes via pHash [Zau10] before running UMAP [MHM20] projection.



(a) *Scatterplot $\mathcal{P}$ (boxes: zoom-ins)* **(b)** *Sca^{2}Gri grid $\mathcal{G}(\mathcal{P})$* **(c)** *Zoom-in z into area with crowns* **(d)** *Show more images via $g_x = 24$*

(e) *Decrease τ_z to reflect $\mathcal{P}$ structure* **(f)** *Focus solid crowns & decrease g_x* **(g)** *Obtain further details by zooming z* **(h)** *Increase g_x for $\mathcal{P}$ structure*

Figure 3: *Sca^{2}Gri exploration example with the 863 610 images from the Droplet dataset. (a) presents the scatterplot from feature extraction via perceptual hashing and UMAP projection. Akin to Fig. 1, the colored boxes correspond to zoom-ins during the exploration (b-h). Gray points depict scatterplot data directly, the images are presented in the grid cells (size is scaled to 80%), and black semitransparent lines denote point-to-cell mapping displacement (d, e & h). Parameters and timings are given at the top of each frame. The data is explored via interactive adjustment of the three Sca^{2}Gri parameters scatterplot area z, grid resolution g_x, and displacement τ_z (updates happen at interactive rates).*

Synthetic Data

For generating synthetic data, we follow a procedure akin to Hilasaca et al. [HMJE*24]. We generate $\|\mathcal{P}\|$ 2D points which are distributed according to a mixture of Gaussians with each Gaussian featuring its own mean and covariance matrix. The distribution of generated points for each Gaussian can be described as follows:

$$p_i \sim \sum_{j=1}^{K} \frac{\|\mathcal{P}\|}{K} \cdot \mathcal{D}(\mu_j, \Sigma_j), \quad \text{for } i = 1, \ldots, \|\mathcal{P}\| \tag{5}$$

where p_i is the i-th 2D point ($p_i \in \mathcal{P}$). Here, K is the number of Gaussian components (randomly sampled with $K \in \{1,2,3,4,5\}$), whereas each Gaussian $\mathcal{D}$ generates $\|\mathcal{P}\|/K$ points. μ_j is the mean of the j-th Gaussian, sampled from a uniform distribution over the range $[0,W] \times [0,H]$ (where $W = 1$ and H is randomly sampled in the range $[0.25,1]$). This means that we vary the width-to-height ratio of the scatterplot domain between 1 and 4. $\Sigma_j = \text{diag}(\sigma^2_{x,j}, \sigma^2_{y,j})$ is the diagonal covariance matrix of the j-th Gaussian, with $\sigma^2_{x,j}$ and $\sigma^2_{y,j}$ sampled randomly.

6. Interactive Exploration

We now exemplify how Sca^{2}Gri can be used for interactive exploration of large data by means of the two real-word datasets (Sec. 5): ImageNet and droplet experiments.

ImageNet

Fig. 1 presents an example exploration of the 1.2 million images in the ImageNet-1k data, showcasing the interactive adjustment of the three Sca^{2}Gri parameters scatterplot area z, grid resolution g_x, and displacement τ. Crucially, all parameter changes are incorporated at highly interactive rates. (a) provides the points of the scatterplot, whereas the colored boxes correspond to zoom-ins during the exploration. In (b–d), we gain an overview on the full data at different grid resolutions g_x. We then aim to have a closer look at an area in the lower right encompassing birds and mammals (orange box), and for this select a zoom box z to yield the new grid in (e). We can see that birds and big cats (especially leopards and tigers) are located closely in the plot, interestingly indicating that similar embeddings are generated for them by ImageNet. We further narrow z for more detailed investigation (blue box in a) and observe visually similar occurrences of snow leopards and owls in particular (f). We discover that they exhibit similarity with respect to facial structures as well as patterns in fur and feathers, respectively.

Finally, we zoom into a different scatterplot area and focus on an outlier structure (red box in a). We observe that the outliers corresponds to series of images taken from different people in front of the same (blueish) background, whereas the other images in the vicinity reflect different motives, mostly incorporating humans (g). From (g) to (i), we increase the displacement τ_z — as reflected by the displacement indicators — to see a larger breadth of images at the cost of increasingly losing the original scatterplot structure.

Droplet experiments

Fig. 3 shows the results of an experimental study on the behavior of droplets falling on a thin film of liquid. In the example exploration, all parameters are adapted (z, τ_z, g_x), and changes are quickly

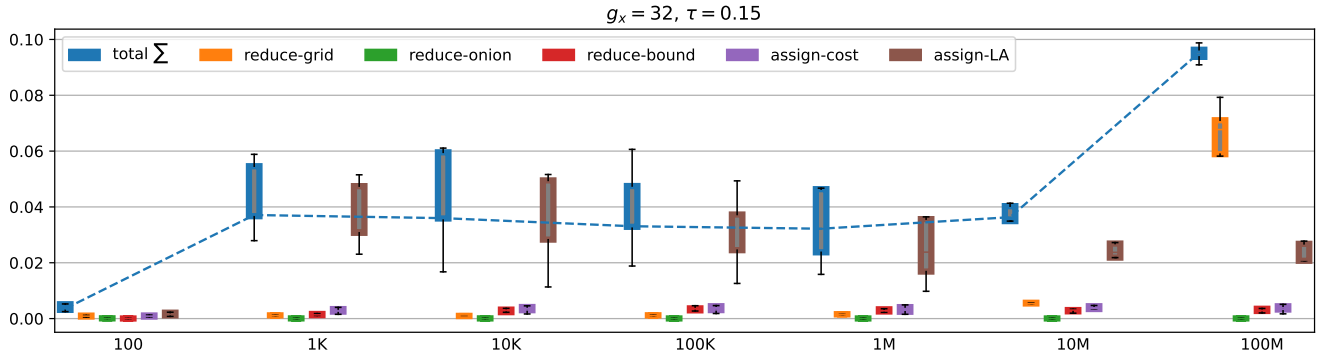


Figure 4: *Scaling properties of Sca²Gri from $\|\mathcal{P}\| = 100$ to $\|\mathcal{P}\| = 100\,000\,000$ (100M). All timings are provided in seconds.*

reflected. (a, b) We start from an overview on the full droplet experiment data that shows the variety of different states in the ensemble. (c) We then focus and zoom into an area that exhibits crown structures of different shapes for closer investigation. (d) To get a better overview of the variety of results in the area, we decrease g_x (e) as well as the maximum resolution threshold τ_z such that the locations of the shown experimental images correspond more closely to their original location in the scatterplot $\mathcal{P}$. (f) We then further zoom into solid (thick) crown structures and decrease g_x to see the images in more detail. (g) We continue by selecting a subarea that lies at the border of different behaviors depicted. (h) Finally, we increase the grid resolution g_x again to see clear separations of clusters of different structures. This view shows that there are several clearly distinguishable local clusters of different experimental images with very close corresponding point positions $\mathcal{P}$. Here, both the local structure of the scatterplot in this area can be preserved and also the various images belonging to individual clusters shown by accordingly choosing the displacement threshold τ_z.

7. Scaling and Parameter Study

We now discuss the results of our scaling and parameter study conducted with 100 synthetically generated scatterplots. In particular, we investigate the performance in $\|\mathcal{P}\| >> \|\mathcal{C}\|$ scenarios which are the focus of this work.

Scaling

We now consider a scaling study for an increasing numbers of points from $\|\mathcal{P}\| = 100$ to $\|\mathcal{P}\| = 100\,000\,000$ (100M) (in practical scenarios the number of cells $\|\mathcal{C}\|$ is bound to thousands at most due to limited visual scalability). The results presented in Fig. 4 show that the (relative) computational footprint of different steps varies depending on $\|\mathcal{P}\|$. For $\|\mathcal{P}\| \leq 10$M, ***assign**-LA* is clearly the most impactful, with ***reduce**-grid* starting to have a noticeable impact for $\|\mathcal{P}\| = 10$M and consuming the largest part of the total compute time for $\|\mathcal{P}\| = 100$M. These two steps also determine the theoretical computational complexity as discussed in Eq. 4.

For $\|\mathcal{P}\| \in [10\text{K}, 10\text{M}]$, the total compute time remains largely constant. Looking at the two main cost factors, we attribute this (i) to the GPU implementation of ***reduce**-grid* which can rapidly process even large point numbers in parallel, and (ii) the fact that the cost of ***assign**-LA* is mainly driven by $\|\mathcal{C}\|$ which remains unchanged. Crucially, while the total time for $\|\mathcal{P}\| = 100$M noticeably increases, the performance is still sufficient for interactive exploration.

Parameter Impact

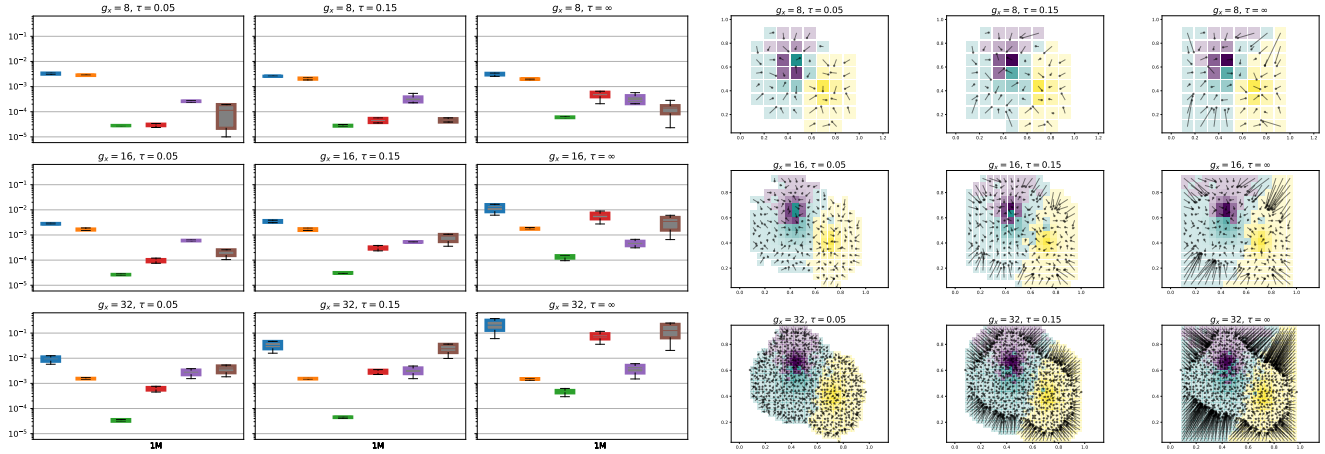
Fig. 5 presents the impact of parameters g_x and τ with $\|\mathcal{P}\| = 1$M. Fig. 5a shows that larger g_x in combination with a more relaxed (larger) displacement threshold τ induces higher cost for (i) ***reduce**-bound* and (ii) ***assign**-LA*. This is due to the fact that both settings have an impact on (i) the number of considered onion layers $\|\mathcal{N}\|$ and (ii) the number of elements e to assign during ***assign**-LA* (see Eq. 2). Note that larger τ means that there is more flexibility in distributing points, with this also increasing the upper bound determined in ***reduce**-bound* and with this eventually the number of reduced points $\|\mathcal{P}^*\|$. Note that complexity-wise, as discussed in Sec. 4.2, $O(\|\mathcal{P}^*\|) = O(\|\mathcal{C}\|)$ regardless of the setting for τ.

The grids in Fig. 5b clearly show that with increasing τ naturally more points can be represented in cells at the cost of more significant displacement (visually indicated by longer gray lines between original point position and cell centers). Similarly, while lower g_x and larger cells sizes s allow to reflect each point in more detail via images of glyphs, it is also detrimental to maintaining structure.

In general, g_x and τ directly influence the fundamental tradeoffs between (i) cell size versus the number of cells (for a fixed domain size) and (ii) the number of represented points versus point displacement. The fact that there is no clear optimal setting and different configurations might be favorable in different phases of the analysis underlines the importance of interactive control, especially for the analysis of scatterplots involving a large number of points.

8. Comparison

While there is no prior technique that accomplishes Sca²Gri's objectives, in this section we aim to put it into context with two recent related methods, namely DGrid and Hagrid, regarding computational performance and quality (Fig. 6). DGrid in particular showed superior performance to other methods in a recent study [HMJE*24]. We employ synthetic scatterplot data (Sec. 5) and use $\tau = \infty$ for Sca²Gri, i.e, no restrictions are placed on displacement.



(a) *Performance in seconds.* **(b)** *Grids with gray lines between points p and assigned cells $\mathcal{G}[p]$.*

Figure 5: *Impact of grid size g_x and displacement τ on (a) performance (box colors akin to Fig. 4: total, **reduce**-grid, **reduce**-onion, **reduce**-bound, **assign**-cost, **assign**-LA) and assignment (b) at the example of synthetic data with $\|\mathcal{P}\|$ =one million points. In (b), cells are colored with respect to the Gaussian that their assigned point stems from. The opacity of the color of each cell $c \in \mathcal{C}$ is further modulated to reflect how many points $\chi_n[c]$ are associated with it: $\alpha(c) = 0.2 + 0.8\frac{\chi_n[c]}{\max_{c' \in \mathcal{C}} \chi_n[c']}$.*

Performance Scaling

Fig. 6a compares the performance scaling of the different methods for increasing numbers of points $\|\mathcal{P}\|$. For Hagrid and DGrid, the grid size is chosen such that the points can be accommodated by the grid as required by these methods, Sca²Gri employs a fixed $g_x = 32$. For low point counts ≤ 10000 points both Hagrid and DGrid are faster than Sca²Gri. This can mainly be attributed to the recursive bijection approach for grid assignment and the use of space-filling curves introduced by DGrid and Hagrid, respectively, which are both computationally faster than the Jonker-Volgenant algorithm we use in Sca²Gri for ***assign**-LA*. However, crucially, while Sca²Gri's timings remain largely constant below 0.1 s up until $m = 10000000$ points, the other method's timings rapidly increase beyond 100 s for 100000 points already.

Quality Metrics

We further evaluate Sca²Gri regarding how well it performs regarding performance metrics in classic scenarios where $\|\mathcal{P}\| < \mathcal{C}$ (Fig. 6b). Note that in such cases, Sca²Gri skips the ***reduce*** phase as discussed in Sec. 4 (due to the fact that no point reduction is possible, yielding $\mathcal{P}^* = \mathcal{P}$). We create 1000 synthetic scatterplots with $\|\mathcal{P}\| \in \{500, \dots, 1000\}$. In addition, also akin to Hilasaca et al. [HMJE*24], we randomly choose a density factor $d \in \{3,5,7,9,11\}$ to control the considered cell size $s = \sqrt{(W \cdot H)/(d \cdot \|\mathcal{P}\|)}$ (with W and H depicting width and height of the scatterplot, respectively).

We consider six comparison metrics as used by Hilasaca et al. [HMJE*24] (from the seven originally employed metrics we omit overlap as there is none by design in DGrid, Hagrid, or Sca²Gri). Note that for DGrid and Hagrid, the numbers in Fig. 6b generally reflect the results reported by Hilasaca et al. [HMJE*24] (Fig. 6).

stress $\in [0, \infty)$ indicates distance preservation (↓ is better):

$$stress = \sqrt{\frac{\sum_{i<j}^{\|} \mathcal{P}\| (\|p_i - p_j\| - \|\mathcal{G}[p_i] - \mathcal{G}[p_j]\|)^2}{\sum_{i<j}^{\|} \mathcal{P}\| \|p_i - p_j\|^2}}$$

There is only minimal *stress* close to zero for both Sca²Gri and DGrid, while Hagrid preserves distances less accurately.

trustworthiness $\in [0,1]$ (↑ is better) quantifies ranking relationships of neighboring points:

$$trustworthiness = 1 - \frac{2}{\|\mathcal{P}\| K (2\|\mathcal{P}\| - 3K - 1)} \sum_{i}^{\|} \mathcal{P}\| \sum_{j \in U_K^i} (r(i,j) - K).$$

Here, U_K^i is the set of points that is in the neighborhood of size K of $\mathcal{G}[p_i] \in \mathcal{C}$ but not in the neighborhood of size K of $p_i \in \mathcal{P}$, and $r(i,j)$ denoes the rank of point p_j in the ordering according to the distance from p_i in the original scatterplot (we use $K = 8$). Hagrid scores still high values around 0.88, whereas close-to-optimal results ≈ 1 are achieved by both Sca²Gri and DGrid.

ordering $\in [0,1]$ indicates order preservation (↓ is better):

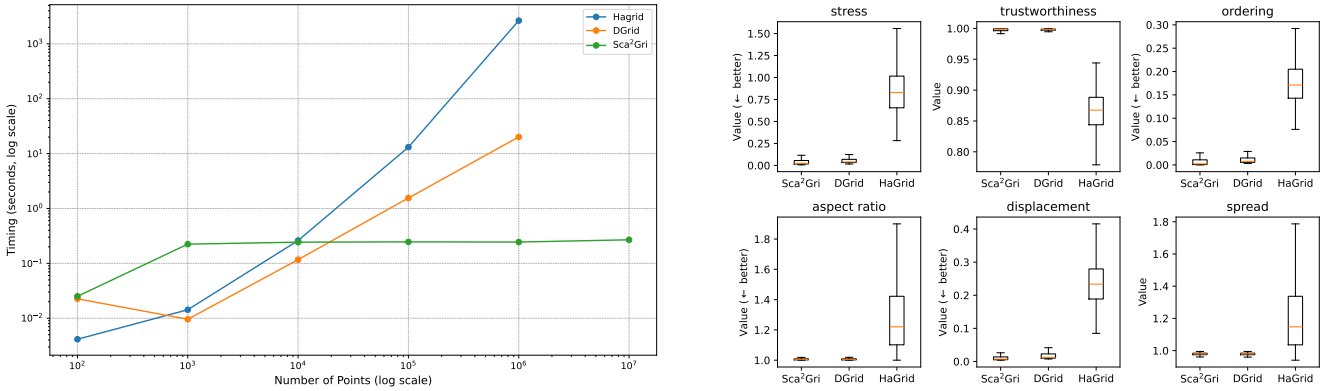
$$ordering = \frac{\sum_{i,j}^{\|} \mathcal{P}\| \left(x_i > x_j \wedge \mathcal{G}[x_i] < \mathcal{G}[x_j] + y_i > y_j \wedge \mathcal{G}[y_i] < y_j' \right)}{\|\mathcal{P}\| (\|\mathcal{P}\| - 1)},$$

with x_i and y_i denoting the x and y coordinates of p_i (likewise for respective cells). Sca²Gri and DGrid demonstrate good order preservation here with values which are close to zero, while Hagrid exhibits more significant deviation.

aspect ratio $\in [1, \infty)$ captures the change in ratio (↓ is better):

$$aspect\ ratio = \max\left(\frac{W' \times H}{H' \times W}, \frac{H' \times W}{W' \times H} \right).$$

W and H denote width and height of the scatterplot, W' and H' the grid extents. The values for Hagrid indicate a noticeable deviation



(a) *Increasing number of points* $\|\mathcal{P}\|$ *(*$g_x = 32$ *and* $\tau =$ *for Sca²Gri)*

(b) *Various metrics used by Hilasaca et al. [HMJE*24]*

Figure 6: *Comparison of Sca²Gri against DGrid [HMJE*24] and Hagrid [CMC*22]. (a) Sca²Gri crucially scales to* $\|\mathcal{P}\| > 1M$ *in contrast to other methods, (b) while also achieving competetitve quality according to several metrics in* $O(\|\mathcal{P}\|) = O(\|\mathcal{C}\|)$ *scenarios.*

of the aspect ratio. In comparison, Sca²Gri and DGrid only cause minimal distortion in this regard.

displacement $\in [0,\infty)$ quantifies the extent of point position changes when assigning them to the grid (↓ is better):

$$displacement = \frac{1}{\|\mathcal{P}\|\sqrt{W' * H'}} \sum_{i}^{\|} \mathcal{P}\|\|p_i - \mathcal{G}[p_i]\|\|.$$

The assignment schemes of Sca²Gri and DGrid only cause minimal point displacement, while this value is significantly larger for Hagrid.

spread depicts the resulting grid size relative to the original extent:

$$spread = \frac{W' \times H'}{W \times H}$$

The generated grid size approximately equals the original extent for Sca²Gri and DGrid, and is slightly larger for Hagrid.

Overall, it can be seen that Sca²Gri performs similarly to DGrid overall and even better than Hagrid with respect to some measures. In Hilasaca et al. [HMJE*24], the authors compare against seven state-of-the-art techniques in total (besides Hagrid [CMC*22] also ReArrange [GPNB17], PFS' [MELS95, HIMF98], VPSC, RWordle-L / C [SSS*12], ProjSnippet [GRP*14], and PRISM [GH10]). They find that "DGrid presents the best trade-off regarding multiple aspects while producing low distortions and bounding the dimensions of the created layouts, consistently resulting in visual representations with readable glyphs" (p. 12), and further showed favorable results in a user study with 51 participants. With this, we conclude that Sca²Gri is not only the first technique for overlap removal in scatterplots that computationally scales to millions of points at interactive rates, but that it also delivers competitive performance in the classically considered scenarios in which $\|\mathcal{C}\| \geq \|\mathcal{P}\|$.

9. Conclusion

Sca²Gri provides a scalable solution for removing overlaps when representing points in scatterplots as glyphs, with a particular focus on dealing with large numbers of points that exceed traditional visual and computational limitations. Specifically, it is designed for cases where the number of points significantly exceeds the amount of glyphs or images that can be effectively displayed—the glyphs would simply be too numerous and their footprint too small when showing all of them. Accounting for this, Sca²Gri displays only a subset of glyphs, and enables flexible user control of (i) cell size, (ii) maximum point-to-cell mapping displacement, and (iii) the focus area in the scatterplot. Crucially, Sca²Gri remains computationally efficient with linear scalability regarding the number of points, and to the best of our knowledge it is the first grid-based method to achieve interactive rates for scatterplots beyond millions of points with overlap-free images.

The paper opens up several directions for future work. Most prominently, our current Python implementation runs all steps besides ***reduce**-grid* on a single CPU core only. Other steps such as ***reduce**-bound* could also be parallelized and utilize multicore CPUs or GPUs for significant speedups. ***assign**-LA* employs a modified Jonker-Volgenant algorithm with no initialization [Cro16]. It optimally solves the linear assignment problem, but faster solutions like Vogel's approximations or the auction algorithm [Ber79] could be considered as well. We anticipate that further improvements in the implementation like these together with out-of-core extensions will potentially allow Sca²Gri to scale even further to billions of points while maintaining interactive performance. In addition, we restrict ourselves to uniform grids in this work, but with minor adjustments in the methods to assign points to cells (***reduce**-grid*) and neighborhood consideration (***reduce**-onion*) also other (potentially unstructured) grids could be handled, e.g. hexagonal grids [CMC*22].

10. Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) - Project Number 327154368 - SFB 1313.

References

[AR88] ANDERSON J. A., ROSENFELD E. (Eds.): *Neurocomputing: Foundations of Research*. MIT Press, Cambridge, MA, USA, 1988. doi:10.1604/9780262010979. 2

[Ber79] BERTSEKAS D. P.: A distributed algorithm for the assignment problem. *Lab. for Information and Decision Systems Working Paper, MIT* (1979). doi:10.1109/CDC.2008.4739098. 10

[BHJS23] BARTHEL K. U., HEZEL N., JUNG K., SCHALL K.: Improved Evaluation and Generation Of Grid Layouts Using Distance Preservation Quality and Linear Assignment Sorting. *Computer Graphics Forum 42*, 1 (2023), 261–276. doi:10.1111/cgf.14718. 2

[Bra14] BRATH R.: 3d infovis is here to stay: Deal with it. In *2014 IEEE VIS International Workshop on 3DVis (3DVis)* (2014), pp. 25–31. doi:10.1109/3DVis.2014.7160096. 2

[BS06] BERTINI E., SANTUCCI G.: Give chance a chance: Modeling density to enhance scatter plot quality through random data sampling. *Information Visualization 5* (2006), 110 – 95. doi:10.1057/palgrave.ivs.9500122. 2

[CGSQ11] CAO N., GOTZ D., SUN J., QU H.: Dicon: Interactive visual analysis of multidimensional clusters. *IEEE Transactions on Visualization and Computer Graphics 17*, 12 (2011), 2581–2590. doi:10.1109/TVCG.2011.188. 1

[CGZ*20] CHEN X., GE T., ZHANG J., CHEN B., FU C.-W., DEUSSEN O., WANG Y.: A recursive subdivision technique for sampling multi-class scatterplots. *IEEE Transactions on Visualization and Computer Graphics 26*, 1 (2020), 729–738. doi:10.1109/TVCG.2019.2934541. 2

[CLNL87] CARR D. B., LITTLEFIELD R. J., NICHOLSON W. L., LITTLEFIELD J. S.: Scatterplot matrix techniques for large n. *Journal of the American Statistical Association 82*, 398 (1987), 424–436. doi:10.1080/01621459.1987.10478445. 2

[CMC*22] CUTURA R., MORARIU C., CHENG Z., WANG Y., WEISKOPF D., SEDLMAIR M.: Hagrid: using hilbert and gosper curves to gridify scatterplots. *Journal of Visualization 25*, 6 (2022), 1291–1307. doi:10.1145/3481549.3481569. 1, 2, 10

[Cro16] CROUSE D. F.: On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems 52*, 4 (2016), 1679–1696. doi:10.1109/TAES.2016.140952. 6, 10

[CYL*21] CHEN C., YUAN J., LU Y., LIU Y., SU H., YUAN S., LIU S.: OoDAnalyzer: Interactive Analysis of Out-of-Distribution Samples. *IEEE Transactions on Visualization and Computer Graphics 27*, 7 (2021), 3335–3349. doi:10.1109/TVCG.2020.2973258. 2

[DBCL87] D. B. CARR R. J. LITTLEFIELD W. L. N., LITTLEFIELD J. S.: Scatterplot matrix techniques for large n. *Journal of the American Statistical Association 82*, 398 (1987), 424–436. doi:10.1080/01621459.1987.10478445. 2

[DMS06] DWYER T., MARRIOTT K., STUCKEY P. J.: Fast node overlap removal. In *Graph Drawing* (Berlin, Heidelberg, 2006), Healy P., Nikolov N. S., (Eds.), Springer Berlin Heidelberg, pp. 153–164. doi:10.1007/11618058_15. 1, 2

[DSF*14] DUARTE F. S. L. G., SIKANSI F., FATORE F. M., FADEL S. G., PAULOVICH F. V.: Nmap: A novel neighborhood preservation space-filling algorithm. *IEEE Transactions on Visualization and Computer Graphics 20*, 12 (Dec 2014), 2063–2071. doi:10.1109/TVCG.2014.2346276. 2

[Elm05] ELMQVIST N.: Balloonprobe: Reducing occlusion in 3d using interactive space distortion. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology* (New York, NY, USA, 2005), VRST '05, Association for Computing Machinery, p. 134–137. doi:10.1145/1101616.1101643. 1, 2

[ENP*09] ELER D. M., NAKAZAKI M. V., PAULOVICH F. V., SANTOS D. P., DE FARIA ANDERY G., DE OLIVEIRA M. C. F., NETO J. B., MINGHIM R.: Visual analysis of image collections. *Vis. Comput. 25*, 10 (2009), 923–937. doi:10.1007/s00371-009-0368-7. 1

[EvSS13] EPPSTEIN D., VAN KREVELD M., SPECKMANN B., STAALS F.: Improved grid map layout by point set matching. In *2013 IEEE Pacific Visualization Symposium (PacificVis)* (2013), pp. 25–32. doi:10.1109/PacificVis.2013.6596124. 2

[FDH*15] FRIED O., DIVERDI S., HALBER M., SIZIKOVA E., FINKELSTEIN A.: IsoMatch: Creating informative grid layouts. *Computer Graphics Forum 34*, 2 (2015), 155–166. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.12549, doi:10.1111/cgf.12549. 2

[Fre22] FREY S.: Optimizing Grid Layouts for Level-of-Detail Exploration of Large Data Collections. *Computer Graphics Forum 41*, 3 (2022), 247–258. doi:10.1111/cgf.14537. 2

[GCM*16] GEPPERT A., CHATZIANAGNOSTOU D., MEISTER C., GOMAA H., LAMANNA G., WEIGAND B.: Classification of Impact Morphology and Splashing/Deposition Limit For N-Hexadecane. *Atomization and Sprays 26*, 10 (2016). doi:10.1615/AtomizSpr.2015013352. 6

[GCNT13] GOMEZ-NIETO E., CASACA W., NONATO L. G., TAUBIN G.: Mixed integer optimization for layout arrangement. In *2013 XXVI Conference on Graphics, Patterns and Images* (2013), pp. 115–122. doi:10.1109/SIBGRAPI.2013.25. 2

[GH10] GANSNER E., HU Y.: Efficient, proximity-preserving node overlap removal. *J. Graph Algorithms Appl. 14* (2010), 53–74. doi:10.7155/jgaa.00198. 1, 2, 10

[GPNB17] GARDEREN M. V., PAMPEL B., NOCAJ A., BRANDES U.: Minimum-Displacement Overlap Removal for Geo-referenced Data Visualization. *Computer Graphics Forum* (2017). doi:10.1111/cgf.13199. 1, 2, 10

[GRP*14] GOMEZ-NIETO E., ROMAN F. S., PAGLIOSA P., CASACA W., HELOU E. S., DE OLIVEIRA M. C. F., NONATO L. G.: Similarity preserving snippet-based visualization of web search results. *IEEE Transactions on Visualization and Computer Graphics 20*, 3 (2014), 457–470. doi:10.1109/TVCG.2013.242. 1, 2, 10

[HIMF98] HAYASHI K., INOUE M., MASUZAWA T., FUJIWARA H.: A layout adjustment problem for disjoint rectangles preserving orthogonal order. In *Graph Drawing* (1998). doi:10.1007/3-540-37623-2_14. 1, 2, 10

[HMJE*24] HILASACA G. M., MARCÍLIO-JR W. E., ELER D. M., MARTINS R. M., PAULOVICH F. V.: A grid-based method for removing overlaps of dimensionality reduction scatterplot layouts. *IEEE Transactions on Visualization and Computer Graphics 30*, 8 (2024), 5733–5749. doi:10.1109/TVCG.2023.3309941. 1, 2, 7, 8, 9, 10

[HMvdW*20] HARRIS C. R., MILLMAN K. J., VAN DER WALT S. J., GOMMERS R., VIRTANEN P., COURNAPEAU D., WIESER E., TAYLOR J., BERG S., SMITH N. J., KERN R., PICUS M., HOYER S., VAN KERKWIJK M. H., BRETT M., HALDANE A., DEL RÍO J. F., WIEBE M., PETERSON P., GÉRARD-MARCHANT P., SHEPPARD K., REDDY T., WECKESSER W., ABBASI H., GOHLKE C., OLIPHANT T. E.: Array programming with NumPy. *Nature 585*, 7825 (2020), 357–362. doi:10.1038/s41586-020-2649-2. 6

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), pp. 770–778. doi:10.1109/CVPR.2016.90. 6

[Joh67] JOHNSON S. C.: Hierarchical clustering schemes. *Psychometrika 32*, 3 (Sept. 1967), 241–254. doi:10.1007/bf02289588. 2

[Koh90] KOHONEN T.: The self-organizing map. *Proceedings of the IEEE 78*, 9 (1990), 1464–1480. doi:10.1109/5.58325. 2

[KSH17] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. *Commun. ACM 60*, 6 (2017), 84–90. doi:10.1145/3065386. 6

[LD15] LIU S., DENG W.: Very deep convolutional neural network based image classification using small training sample size. In *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)* (2015), pp. 730–734. doi:10.1109/ACPR.2015.7486599. 6

[LMvW10] LI J., MARTENS J., VAN WIJK J. J.: A model of symbol size discrimination in scatterplots. In *Proceedings of the 28th International Conference on Human Factors in Computing Systems* (2010). doi:10.1145/1753326.1753714. 2

[LPS15] LAM S. K., PITROU A., SEIBERT S.: Numba: A llvm-based python jit compiler. In *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC* (2015), pp. 1–6. doi:10.1145/2833157.2833162. 6

[LSL*23] LI Z., SHI R., LIU Y., LONG S., GUO Z., JIA S., ZHANG J.: Dual space coupling model guided overlap-free scatterplot. *IEEE Transactions on Visualization and Computer Graphics 29*, 1 (2023), 657–667. doi:10.1109/TVCG.2022.3209459. 2

[LvM09] LI J., VAN WIJK J. J., MARTENS J.: Evaluation of symbol contrast in scatterplots. In *2009 IEEE Pacific Visualization Symposium* (2009). doi:10.1109/PACIFICVIS.2009.4906843. 2

[MDS*17] MEULEMANS W., DYKES J., SLINGSBY A., TURKAY C., WOOD J.: Small multiples with gaps. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 381–390. doi:10.1109/tvcg.2016.2598542. 2

[MELS95] MISUE K., EADES P., LAI W., SUGIYAMA K.: Layout adjustment and the mental map. *J. Vis. Lang. Comput. 6* (1995), 183–210. doi:10.1006/jvlc.1995.1010. 1, 2, 10

[MG13a] MAYORGA A., GLEICHER M.: Splatterplots: Overcoming overdraw in scatter plots. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1526–1538. doi:10.1109/TVCG.2013.65. 1, 2

[MG13b] MAYORGA A., GLEICHER M.: Splatterplots: Overcoming overdraw in scatter plots. *IEEE Transactions on Visualization and Computer Graphics 19*, 9 (2013), 1526–1538. doi:10.1109/TVCG.2013.65. 2

[MH17] MCNEILL G., HALE S. A.: Generating tile maps. *Computer Graphics Forum 36*, 3 (2017), 435–445. doi:https://doi.org/10.1111/cgf.13200. 2

[MHM20] MCINNES L., HEALY J., MELVILLE J.: Umap: Uniform manifold approximation and projection for dimension reduction, 2020. arXiv:1802.03426. 1, 6

[MPOW17] MICALLEF L., PALMAS G., OULASVIRTA A., WEINKAUF T.: Towards perceptual optimization of the visual design of scatterplots. *IEEE Transactions on Visualization and Computer Graphics* (2017). doi:10.1109/TVCG.2017.2674978. 1, 2

[MSS21] MEULEMANS W., SONDAG M., SPECKMANN B.: A Simple Pipeline for Coherent Grid Maps. *IEEE Transactions on Visualization and Computer Graphics 27*, 2 (2021), 1236–1246. doi:10.1109/TVCG.2020.3028953. 2

[NNB*16] NACHMANSON L., NOCAJ A., BEREG S., ZHANG L., HOLROYD A.: Node overlap removal by growing a tree. In *Graph Drawing and Network Visualization* (Cham, 2016), Hu Y., Nöllenburg M., (Eds.), Springer International Publishing, pp. 33–43. doi:10.1007/978-3-319-50106-2_3. 1, 2

[Pd09] PINHO R., DE OLIVEIRA M. C. F.: Hexboard: Conveying pairwise similarity in an incremental visualization space. In *2009 13th International Conference Information Visualisation* (2009), pp. 32–37. doi:10.1109/IV.2009.12. 1, 2

[PdOdAL09] PINHO R., DE OLIVEIRA M. C. F., DE A. LOPES A.: Incremental board: A grid-based space for visualizing dynamic data sets. In *Proceedings of the 2009 ACM Symposium on Applied Computing* (New York, NY, USA, 2009), Association for Computing Machinery, p. 1757–1764. doi:10.1145/1529282.1529679. 1, 2

[PSSC17] PAHINS C. A. L., STEPHENS S. A., SCHEIDEGGER C., COMBA J. L. D.: Hashedcubes: Simple, low memory, real-time visual exploration of big data. *IEEE Transactions on Visualization and Computer Graphics 23*, 1 (2017), 671–680. doi:10.1109/TVCG.2016.2598624. 2

[QSST10] QUADRIANTO N., SMOLA A. J., SONG L., TUYTELAARS T.: Kernelized sorting. *IEEE Transactions on Pattern Analysis and Machine Intelligence 32*, 10 (Oct 2010), 1809–1821. doi:10.1109/TPAMI.2009.184. 2

[Rai34] RAISZ E.: The rectangular statistical cartogram. *Geographical Review 24* (1934), 292–296. doi:doi.org/10.2307/208794. 2

[SG11] STRONG G., GONG M.: Data organization and visualization using self-sorting map. In *Proceedings of Graphics Interface 2011* (2011), GI '11, p. 199–206. doi:10.1111/cgf.12549. 2

[SG18] SARIKAYA A., GLEICHER M.: Scatterplots: Tasks, data, and designs. *IEEE Transactions on Visualization and Computer Graphics 24*, 1 (2018), 402–412. doi:10.1109/TVCG.2017.2744184. 1, 2

[SMT13] SEDLMAIR M., MUNZNER T., TORY M.: Empirical guidance on scatterplot and dimension reduction technique choices. *IEEE Transactions on Visualization and Computer Graphics 19*, 12 (2013), 2634–2643. doi:10.1109/TVCG.2013.153. 1, 2

[SSS*12] STROBELT H., SPICKER M., STOFFEL A., KEIM D., DEUSSEN O.: Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives. *Computer Graphics Forum 31* (2012). doi:10.1111/j.1467-8659.2012.03106.x. 2, 10

[TSW*07] TORY M., SPRAGUE D., WU F., SO W. Y., MUNZNER T.: Spatialization design: Comparing points and landscapes. *IEEE Transactions on Visualization and Computer Graphics 13*, 6 (2007), 1262–1269. doi:10.1109/TVCG.2007.70596. 2

[VGO*20] VIRTANEN P., GOMMERS R., OLIPHANT T. E., HABERLAND M., REDDY T., COURNAPEAU D., BUROVSKI E., PETERSON P., WECKESSER W., BRIGHT J., VAN DER WALT S. J., BRETT M., WILSON J., MILLMAN K. J., MAYOROV N., NELSON A. R. J., JONES E., KERN R., LARSON E., CAREY C. J., POLAT İ., FENG Y., MOORE E. W., VANDERPLAS J., LAXALDE D., PERKTOLD J., CIMRMAN R., HENRIKSEN I., QUINTERO E. A., HARRIS C. R., ARCHIBALD A. M., RIBEIRO A. H., PEDREGOSA F., VAN MULBREGT P., SCIPY 1.0 CONTRIBUTORS: SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods 17* (2020), 261–272. doi:10.1038/s41592-019-0686-2. 6

[vKS04] VAN KREVELD M., SPECKMANN B.: On rectangular cartograms. In *Algorithms – ESA 2004* (Berlin, Heidelberg, 2004), Albers S., Radzik T., (Eds.), Springer Berlin Heidelberg, pp. 724–735. doi:10.1016/j.comgeo.2006.06.002. 2

[WD08] WOOD J., DYKES J.: Spatially Ordered Treemaps. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov. 2008), 1348–1355. doi:10.1109/TVCG.2008.165. 2

[WDS10] WOOD J., DYKES J., SLINGSBY A.: Visualisation of origins, destinations and flows with od maps. *The Cartographic Journal 47* (2010), 117 – 129. doi:10.1179/000870410X12658023467367. 2

[YXX*21] YUAN J., XIANG S., XIA J., YU L., LIU S.: Evaluation of sampling methods for scatterplots. *IEEE Transactions on Visualization and Computer Graphics 27*, 02 (2021), 1720–1730. doi:10.1109/TVCG.2020.3030432. 1, 2

[Zau10] ZAUNER C.: Implementation and Benchmarking of Perceptual Image Hash Functions. 6

[ZYC*24] ZHOU Y., YANG W., CHEN J., CHEN C., SHEN Z., LUO X., YU L., LIU S.: Cluster-Aware Grid Layout. *IEEE Transactions on Visualization and Computer Graphics 30*, 1 (2024), 240–250. doi:10.1109/TVCG.2023.3326934. 2