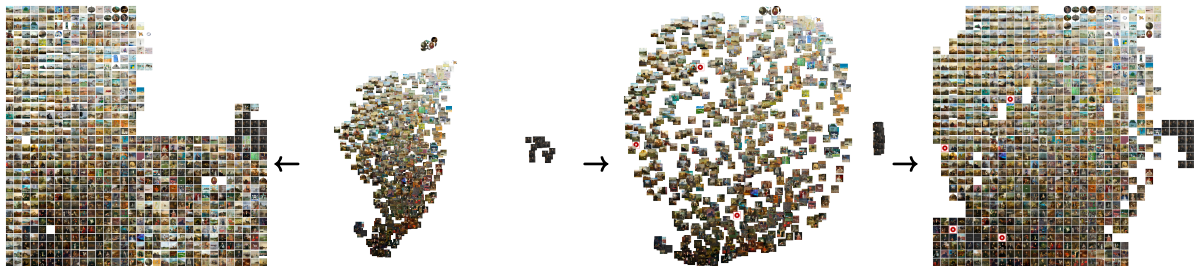
SiGrid: Gridifying Scatterplots with Sector-Based Regularization and Hagrid

Rene Cutura* University of Stuttgart
Hennes Rave† University of Münster
Quynh Quang Ngo‡ University of Stuttgart
Vladimir Molchanov§ University of Münster
Lars Linsen¶ University of Münster
Daniel Weiskopf‖ University of Stuttgart
Michael Sedlmair** University of Stuttgart



(a) Gridified scatterplot with Hagrid [2]

(b) Input scatterplot

(c) Regularized scatterplot by SBR [14]

(d) Our gridified scatterplot with SiGrid

Fig. 1: (b) Original scatterplot of an image dataset [1], where the images are plotted at the point locations. The scatterplot exhibits a lot of overdraw, making it hard to grasp the content of the images. (a) Gridified scatterplot obtained with the state-of-the-art method Hagrid [2]. (c) Regularized scatterplot after only 1 iteration of Sector-Based Regularization [14]. (d) Gridified scatterplot of our technique SiGrid that takes the output (c) and gridifies it with Hagrid. The comparison between Hagrid (a) and our new SiGrid technique (d) reveals that the global structures from the input scatterplot (b) are presented better in (d). For example, the decreasing brightness trend from top to bottom in (b) is reflected better in (d) than in (a), where the bottom-left and bottom-right corners of (a) are bright, whereas the bottom part of (b) is dark.

Abstract

Hagrid is a state-of-the-art space-filling-curve-based method for gridifying scatterplots. However, it exhibits limitations in preserving the global structures of scatterplots with areas of varying density due to the incompatibility of adapting the granularity level of the underlying space-filling curve to regions with different densities. To compensate for this shortcoming, we introduce SiGrid, which combines Hagrid with the Sector-Based Regularization (SBR) technique. SiGrid applies SBR to generate a scatterplot with a more uniform and generally lower density as an intermediate step. This intermediate scatterplot can then be fed to Hagrid for improved results. We quantitatively evaluate SiGrid by comparing it to Hagrid over a set of 502 scatterplots of different sizes, ranging from 50 to 10 000 points per dataset, using relevant quality metrics. While generally slower, the results demonstrate that SiGrid outperforms Hagrid regarding the quality metrics of rank-wise neighborhood preservation (trustworthiness), ordering preservation, and pairwise distance preservation (cross-correlation).

Index Terms: Scatterplot, Space-filling curve, Grid layout, Neighborhood preservation

*e-mail: rene.cutura@visus.uni-stuttgart.de
†e-mail: hennes.rave@uni-muenster.de
‡e-mail: quynh.ngo@visus.uni-stuttgart.de
§e-mail: molchano@uni-muenster.de
¶e-mail: linsen@uni-muenster.de
‖e-mail: daniel.weiskopf@visus.uni-stuttgart.de
**e-mail: michael.sedlmair@visus.uni-stuttgart.de

1 Introduction

How to organize a scatterplot of data with icons or glyphs [4, 17] into an aesthetic and easy-to-enrich layout (e.g., with the icons or glyphs) has recently gained much research attention [2, 6, 10, 14, 15]. A typical application is in the context of Dimensionality Reduction (DR), where the objective is to maintain the structures of the original dataset as derived and revealed by the DR method, while arranging the display to achieve the desired layout [6, 10, 15]. Existing techniques commonly rearrange the scatterplot by de-cluttering it, e.g., by using Integral Images [15] or Sector-Based Regularization (SBR) [14], or further gridifying it, e.g., by applying DGrid [6] or Hagrid [2]. These techniques make the scatterplot easier to read and enrich it with more intuitive visual encodings, e.g., the scatterplot points can be replaced by pictures (or icons, glyphs) to represent the data items given a gridified layout, see Fig. 1.

Among these post-processing techniques for scatterplots, the state-of-the-art technique Hagrid [2] rearranges the input scatterplot into a regular grid of points without overlap, where the sizes of the embedded glyphs or images can be uniform and controlled. The main idea behind Hagrid is to apply a space-filling curve (SFC) technique [7, 18] on the 2D domain to rearrange the scatterplot points along the curve. The SFC is a recursive structure, where every grid cell gets divided into multiple grid cells after each recursion, which increases the granularity of the given grid. In some cases, multiple points could be assigned to the same grid cell. Hagrid handles those collisions by moving the points along the SFC to the next free spot, see Fig. 2. However, Hagrid exhibits some limitations in preserving the structure of the scatterplots with dense areas or regions of varying density [2]. Too many regions with different densities make it difficult to achieve a regular grid output. One could address this problem by selectively going more granular in a spot (as it affects the whole area), but then the output would not be a regular

grid anymore, and the affected grid cell sizes would become smaller.

In this paper, we propose SiGrid as an improved extension of Hagrid for gridifying scatterplots by combining SBR [14] with Hagrid [2]. SBR transforms scatterplots by iteratively moving points away from dense regions and toward sparse regions, resulting in a more uniform distribution while largely preserving neighborhood relationships. These properties are precisely what Hagrid needs from its input to circumvent the limitation mentioned above. Therefore, we propose running Hagrid on the output of SBR to improve its quality. By limiting SBR to only a few iterations, we can largely avoid its main drawbacks: long runtime and artifacts near the domain boundaries.

We evaluate our approach quantitatively by comparing it with the original Hagrid approach regarding runtime performance and relevant quality metrics over a set of scatterplots of various sizes. The results show that, while being slower, SiGrid improves the output quality when compared with Hagrid. In particular, SiGrid tends to be better than Hagrid regarding rank-wise neighborhood preservation (trustworthiness), ordering preservation, and pairwise distance preservation (cross-correlation) metrics. SiGrid also reduces the number of collisions when compared with Hagrid.

In summary, we contribute SiGrid, an improved method of Hagrid in gridifying scatterplots, which enhances Hagrid by feeding it with regularized scatterplots generated by SBR. A JavaScript implementation of SiGrid is available at https://github.com/saehm/SiGrid.

2 Related Work

Scatterplots are a fundamental tool for visualizing the outcomes of DR techniques, enabling analysts to identify clusters, outliers, and trends within high-dimensional data. However, as datasets grow in size and complexity, scatterplots often suffer from overplotting and visual clutter, which can obscure meaningful patterns. To enhance the interpretability of scatterplots, interactive techniques have been developed. Star Coordinates (SC) is one such method that allows users to manipulate the layout of scatterplots interactively, facilitating sensitivity analysis by adjusting the influence of each dimension [8, 9]. Molchanov and Linsen [11] introduced an approach for the interactive design of multidimensional data projection layouts, enabling users to steer the relative positions of clusters directly within the projection space. Building upon this, they proposed shape-preserving conditions for SC [12], ensuring that the intrinsic structure of the data is maintained during interactive transformations.

Recently, Machado et al. [10] presented Shape-Regularized Projections (ShaRP), a technique that provides users with explicit control over the visual signature of scatterplots. ShaRP allows for the shaping of clusters in both 2D and 3D projections, balancing user-defined layout constraints with the preservation of data relationships.

Visual clutter remains a significant challenge in scatterplots, particularly with large datasets. Ellis and Dix [3] developed a taxonomy categorizing various strategies for clutter reduction in information visualization, including techniques like sampling, aggregation, and dimensional stacking. Rave et al. [15] introduced a method to declutter scatterplots by applying a smooth spatial transformation that regularizes sample distribution while preserving neighborhood relations. This approach enhances the readability of dense scatterplots and has been further refined to achieve uniform sample distribution through SBR [14].

In scenarios where data points are represented by glyphs, icons, or pictograms, such as image collections, occlusion becomes a more pressing issue due to the fixed screen space required for each glyph. To address this, several grid-based methods have been proposed. Hilasaca et al. [6] introduced DGrid, a post-processing strategy that removes overlaps in DR layouts by preserving the original layout's characteristics and bounding the minimum glyph sizes. DGrid adds dummy points to the layout to preserve the whitespace between clusters as best as possible. Depending on the parameter Δ, and the width and the height of the glyphs, DGrid decides the number of dummy points to be added. Due to the dependence of those parameter settings on the data, providing default parameters is impossible. Thus, hand-picking the parameters for an optimal layout is the best choice for using DGrid. With SiGrid, it is possible to provide default parameters. The evaluation shows, see Fig. 5, that only a few iterations of SBR before using Hagrid with default parameterization lead to an improvement.

Raidou et al. [13] proposed a pixel-based mapping technique to relax dense scatterplots, improving the visibility of individual data points. Cutura et al. [2] developed Hagrid, which employs SFCs like Hilbert and Gosper curves to gridify scatterplots efficiently, offering faster execution times. Preservation of spatial relationships among data points (ordering, neighborhoods) highly depends on the SFC resolution and data properties. Recently, Frey [5] proposed reducing the number of points before assigning them to cells in the gridified scatterplot, addressing the scalability issue when the number of points is larger than of available cells.

Improving the readability of scatterplots by reducing visual clutter is increasingly important as data sizes continue to grow. The existing spatial transformations in the visual domain aim to make effective use of available screen space, but often face limitations related to violating neighborhood relationships. Our approach addresses these issues by minimizing information loss in the optimized layout (i.e., cluster preservation). We achieve this through the consistent application of smooth regularization of SBR [14] and the gridification of Hagrid [2], which removes occlusion of glyphed data in scatterplots.

3 SiGrid

This section presents how we combine SBR with Hagrid to improve its visualization qualities. Our main idea is to compensate for the limitation of Hagrid by feeding it an intermediate scatterplot regularized by SBR [14]. First, we provide a brief review of Hagrid and a description of its limitations. We then review the SBR technique and justify combining the method with Hagrid to form SiGrid. The composition of SiGrid's components is then presented.

3.1 Hagrid

Cutura et al. [2] proposed Hagrid by applying SFC to rearrange the points of a scatterplot in a grid-like manner to make it easier to superimpose the original images (or glyphs) directly on the scatterplot's points. The main idea of the approach is illustrated in Fig. 2 (left), where data points move to the center of grid cells. Multiple data points in one cell form a collision, which is resolved by arranging points along the SFC. Figure 2 (right) provides a "bad case" example for Hagrid, where the incompatibility of the current level of the SFC with the points' density leads to many collisions that the method has to resolve. As a consequence, the structure of the points, such as the neighborhood, is not preserved well.

The runtime complexity of Hagrid [2] in the best case is $O(N \log N)$, and $O(N^2 \log N)$ in the worst case, where N is the number of points in the scatterplot.

3.2 Sector-Based Regularization

From our empirical observations, the decluttering techniques SBR and Integral Image (InIm) [15] are superb at lessening the scatterplots' density, transforming scatterplots to become less dense after several iterations.

The idea of the InIm approach is to transform the scatterplot into a density field by quantifying global density using two-dimensional prefix sums (integral images) in 8 orientations. This global density distribution is then used to move points away from overpopulated and toward underpopulated regions. The SBR [14] approach follows up on the InIm idea and generalizes it. It quantifies global density by dividing the domain into sectors around each point, counting

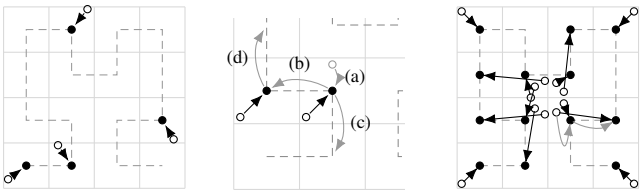
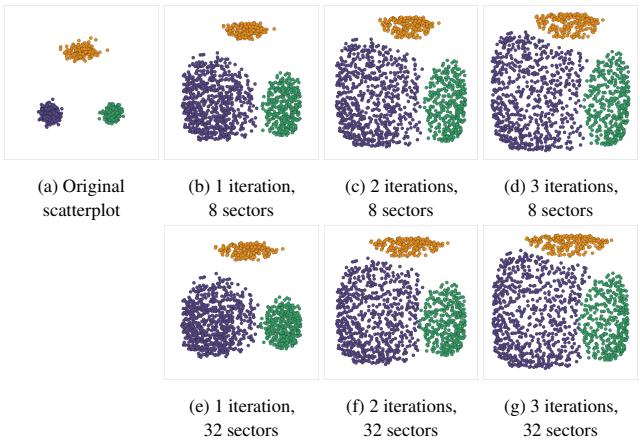


Fig. 2: Points (hollow) getting assigned to grid cells (filled). *(left)* The points are assigned to their grid cells one after another. *(center)* If a point (gray) gets assigned to a grid cell already occupied (a), Hagrid resolves that collision by moving the point in the direction (b) or (c), depending on which one is closer to the input point. If that spot is also already occupied, it continues in that direction to find a free spot (d). *(right)* Many points are assigned to the same cells in the middle of the grid, which leads to many collisions. For example, the (hollow) point in the bottom right has to jump twice (indicated by gray arrows) to get the final arrangement (indicated with the black arrow).

the points in each sector, and dividing by the sector's area. It has a worse runtime complexity of $O(N^2)$, but the number of sectors can be controlled, increasing angular resolution. This leads to a better regularization quality than InIm [14, 15] in terms of having a more uniformly distributed scatterplot. Figure 3 provides the outputs of SBR for one example scatterplot with different numbers of sectors. We observe that the scatterplot's density is lower and more uniform already after a few iterations.



(a) Original scatterplot
(b) 1 iteration, 8 sectors
(c) 2 iterations, 8 sectors
(d) 3 iterations, 8 sectors
(e) 1 iteration, 32 sectors
(f) 2 iterations, 32 sectors
(g) 3 iterations, 32 sectors

Fig. 3: The results of SBR with different parameter settings.

Lessening the density and a more uniform distribution of the scatterplot are precisely what Hagrid needs from its input to circumvent the limitation. We, therefore, apply Hagrid to the output of the SBR [14] technique.

3.3 Composition of SiGrid Components

Given a scatterplot, we obtain the output of SiGrid in two steps as detailed below. The data flow of SiGrid is illustrated in Fig. 4 for a particular scatterplot and compared to the output obtained by Hagrid.

SBR step: We run the sector-based regularization for a few iterations to the input scatterplot. Figure 4c presents the intermediate scatterplot after 3 iterations for the input shown in Fig. 4b.

Hagrid step: The intermediate scatterplot from the first step is gridified by applying Hagrid. Figure 4d shows the final output of our approach for the input scatterplot.

We observe that Fig. 4d better represents the structures of the input scatterplot when compared to the output from applying directly Hagrid in Fig. 4a. For example, the purple group in the upper-left corner and the bottom green cluster (in the input in Fig. 4b) stay as groups in Fig. 4d but get separated in Fig. 4a.

3.4 Runtime

The runtime complexity of Hagrid [2] is $O(N \log N)$ in the best case and $O(N^2 \log N)$ in the worst case, while that of SBR [14] is $O(N^2)$, with N being the number of points in the scatterplot. Hagrid can degrade to $O(N^2 \log N)$ if every point collides, which can be avoided in SiGrid by using SBR as a step preceding Hagrid (see Fig. 5 for the evaluation of collisions). Therefore, SiGrid is $O(N^2)$, as derived from the subsequent execution of SBR in $O(N^2)$.

4 Evaluation

We evaluate SiGrid quantitatively by comparing it with Hagrid over 502 scatterplots with sizes ranging from 50 to 10000 points per dataset (which were also used in the original Hagrid paper [2]), regarding runtime performance and relevant quality metrics described in the following.

4.1 Metrics

In our context, both Hagrid and SiGrid are mappings from $\mathbb{R}^2$ to $\mathbb{R}^2$. We propose using the rank-wise neighborhood preservation (trustworthiness), ordering preservation, and pairwise distances preservation (cross-correlation) metrics to compare the two methods.

Trustworthiness. Trustworthiness [16] is a distance rank-based measure used to measure how well a data transformation (mapping from one metric space to another) preserves the rank-wise neighborhood. In this paper, we choose $k = \lfloor N \cdot 0.05 \rfloor$ as the neighborhood size for trustworthiness.

Ordering. Used in DGrid [6] in evaluation for gridifying techniques, the ordering measure OO^x counts how many points are not in the right order after being mapped, on the x-axis, OO^y on the y-axis, i.e.,

$$OO^x = \sum_{i,j}^{N} \begin{cases} 1 & \text{if} \quad x_i > x_j \wedge x'_i < x'_j \\ 0 & \text{otherwise} \end{cases}$$

$$OO^y = \sum_{i,j}^{N} \begin{cases} 1 & \text{if} \quad y_i > y_j \wedge y'_i < y'_j \\ 0 & \text{otherwise} \end{cases}$$

$$OO = 1 - \frac{1}{N(N-1)} \left(OO^x + OO^y\right),$$

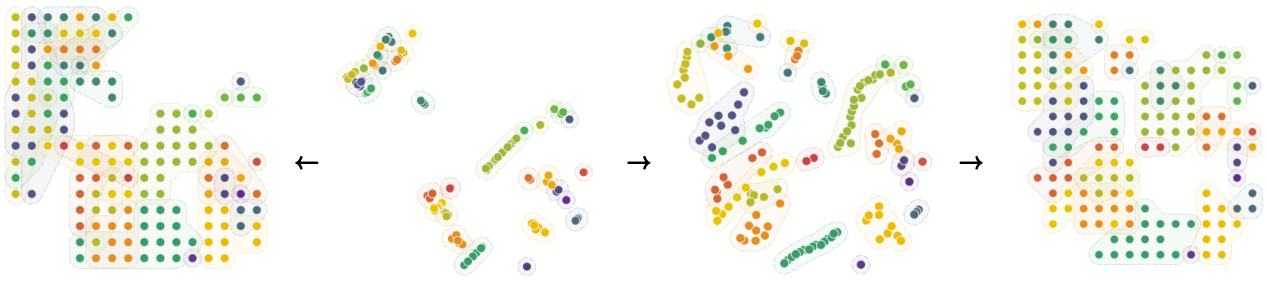
where x_i, y_i are x-, y-coordinates of the input points respectively, and x'_i, y'_i are the their output through the mapping.

Cross-Correlation (*CC*). Also used in DGrid [6], *CC* measures pairwise distances preservation of the mappings. The *CC* measure [2] is defined as:

$$CC = \sum_{i=1}^{N} \sum_{j=1}^{N} \frac{\left(\delta^{\mathbf{Y}}_{i,j} - \overline{\delta^{\mathbf{Y}}}\right) \cdot \left(\delta^{\mathbf{X}}_{i,j} - \overline{\delta^{\mathbf{X}}}\right)}{\sigma_X \cdot \sigma_Y},$$

where $\delta^{\mathbf{X}}_{i,j}$ is the distance between two points x_i and x_j in the original set of points X, and $\delta^{\mathbf{Y}}_{i,j}$ is of the two points y_i and y_j in the gridified set of points Y. The σ_X and σ_Y are the respective standard deviations, and $\overline{\delta^{\mathbf{X}}}$ and $\overline{\delta^{\mathbf{Y}}}$ are the respective mean distances between points.

Furthermore, we also provide two more metrics related to the overall measured runtime evaluation, namely *Collisions* and *Duration*. **Collisions** count how often a point in the input scatterplot gets assigned to an already occupied spot. The smaller this number, the faster the process (for the techniques) and the better the quality (as



(a) Only gridified (b) Projection (c) Inflated (d) Gridified

Fig. 4: (a) The original gridifying technique Hagrid [2], (b) the input scatterplot, and the steps of our technique SiGrid: (c) SBR [14] step and (d) subsequent gridification via Hagrid [2].

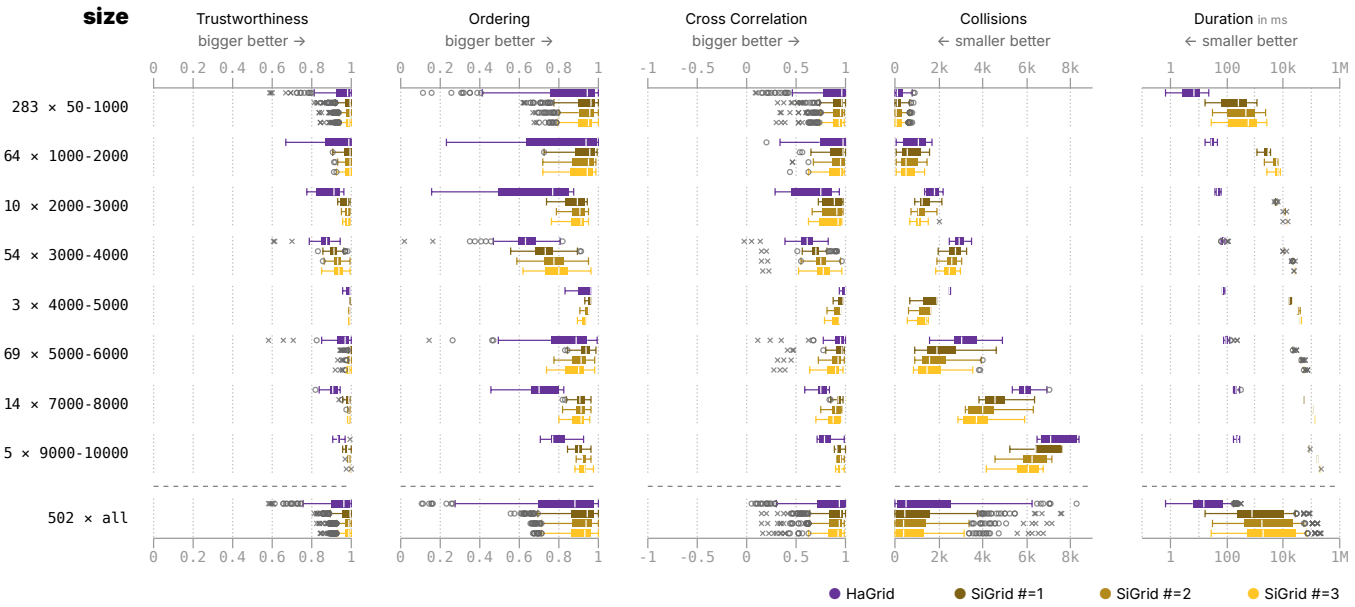


Fig. 5: Comparison of Hagrid and SiGrid (with 1, 2, or 3 SBR iterations). Datasets are binned by their number of points. Except for the 4 000–5 000 bin, SiGrid leads to an improvement in trustworthiness, ordering, cross-correlation, and collisions at the cost of increased runtime. (Hagrid parameters are level: min_level, ensure_whitespace: 0; SBR parameters are iterations: 1, 2, or 3, number_of_sectors: 8). Hagrid needs to use a finer grid for datasets with $N > 4\,096$ to be able to assign all points, which leads to 16 384 grid cells.

the points are not moved around too much). **Duration** is the overall measured runtime (in milliseconds) for each technique running on a dataset. In this paper, the computations are executed on one core of a cluster workstation with AMD EPYC 7662 for all the datasets and the two techniques.

4.2 Results

Figure 5 shows the descriptive statistics of Hagrid and SiGrid in the form of boxplots for each metric, running over each subset of the 502 scatterplots, which are binned by their number of points. We also provide the evaluation for SiGrid for varying number of iterations (1, 2, or 3) in the SBR step to generate the intermediate scatterplot (while keeping other parameter settings fixed).

Overall, SiGrid outperforms Hagrid regarding the quality metrics at the cost of lower runtime performance. Figure 5 shows a clear trend that SiGrid has higher quality with respect to Trustworthiness. Also, for Ordering and CC, SiGrid outperforms Hagrid, except for just 3 scatterplots with the number of points in the range of 4000 to 5000. For all the scatterplot subsets, Collisions for SiGrid are smaller than for Hagrid. However, SiGrid falls behind Hagrid regarding the runtime performance by two orders of magnitude (shown in the Duration column of Fig. 5).

Furthermore, increasing the number of iterations in the SBR step also improves the quality of SiGrid. The Collisions also decrease when increasing the number of iterations in the step, see the Collisions column of Fig. 5. On the other hand, the Duration metric obviously increases when the number of iterations is increased. These measurements indicate that the regularization step supports the performance of the gridifying step in SiGrid.

5 Conclusion

We have presented SiGrid, an improved method of Hagrid for gridifying scatterplots by combining it with the SBR method. While Hagrid is recommended when interactive speed is required, SiGrid is superior for difficult datasets that lead to low gridification quality in Hagrid. We plan to improve the runtime performance of SiGrid in future work. We also plan to provide a more extensive evaluation including all possible technique choices (Hagrid [2], DGrid [6], SBR [14], and InIm-based method [15]).

Acknowledgments

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID 251654672 – TRR 161, project A08, Project-ID 431460824 – CRC 1450, project Z01, DFG project MO 3050/2-3 – 360330772, and under Germany's Excellence Strategy – EXC 2120/1 – 390831618.

References

[1] E. Crowley and A. Zisserman. The State of the Art: Object Retrieval in Paintings using Discriminative Regions. In *British Machine Vision Conference*. BMVA Press, 2014. doi: 10.5244/C.28.38 1

[2] R. Cutura, C. Morariu, Z. Cheng, Y. Wang, D. Weiskopf, and M. Sedlmair. Hagrid: Using Hilbert and Gosper Curves to Gridify Scatterplots. vol. 25, pp. 1291–1307. Springer, 2022. doi: 10.1007/s12650-022-00854-7 1, 2, 3, 4

[3] G. Ellis and A. Dix. A Taxonomy of Clutter Reduction for Information Visualisation. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1216–1223, 2007. doi: 10.1109/TVCG.2007.70535 2

[4] M. Espadoto, R. M. Martins, A. Kerren, N. S. T. Hirata, and A. C. Telea. Toward a Quantitative Survey of Dimension Reduction Techniques. *IEEE Transactions on Visualization and Computer Graphics*, 27(3):2153–2173, 2021. doi: 10.1109/TVCG.2019.2944182 1

[5] S. Frey. Sca2Gri: Scalable Gridified Scatterplots. *Computer Graphics Forum*, 2025. doi: 10.1111/cgf.70141 2

[6] G. M. Hilasaca, W. E. Marcílio-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*, 30(8):5733–5749, 2024. doi: 10.1109/TVCG.2023.3309941 1, 2, 3, 4

[7] D. Hilbert. Über die stetige Abbildung einer Linie auf ein Flächenstück. In *Dritter Band: Analysis · Grundlagen der Mathematik · Physik Verschiedenes*, pp. 1–2. Springer, 1935. doi: 10.1007/978-3-662-38452-7_1 1

[8] E. Kandogan. Star Coordinates: A Multi-dimensional Visualization Technique with Uniform Treatment of Dimensions. In *IEEE Information Visualization Symposium*, pp. 4–8, 2000. 2

[9] E. Kandogan. Visualizing Multi-dimensional Clusters, Trends, and Outliers using Star Coordinates. In *ACM International Conference on Knowledge Discovery and Data Mining*, pp. 107–116. ACM, 2001. doi: 10.1145/502512.502530 2

[10] A. Machado, A. Telea, and M. Behrisch. Controlling the Scatterplot Shapes of 2D and 3D Multidimensional Projections. *Computers & Graphics*, 124(C), 2024. doi: 10.1016/j.cag.2024.104093 1, 2

[11] V. Molchanov and L. Linsen. Interactive Design of Multidimensional Data Projection Layout. In *EuroVis - Short Papers*, pp. 25–29, 2014. doi: 10.2312/eurovisshort.20141152 2

[12] V. Molchanov and L. Linsen. Shape-preserving Star Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):449–458, 2019. doi: 10.1109/TVCG.2018.2865118 2

[13] R. G. Raidou, M. E. Gröller, and M. Eisemann. Relaxing Dense Scatter Plots with Pixel-Based Mappings. *IEEE Transactions on Visualization and Computer Graphics*, 25(6):2205–2216, 2019. doi: 10.1109/TVCG.2019.2903956 2

[14] H. Rave, V. Molchanov, and L. Linsen. Uniform Sample Distribution in Scatterplots via Sector-based Transformation. In *IEEE Visualization and Visual Analytics (VIS)*, pp. 156–160, 2024. doi: 10.1109/VIS55277.2024.00039 1, 2, 3, 4

[15] H. Rave, V. Molchanov, and L. Linsen. De-Cluttering Scatterplots With Integral Images. *IEEE Transactions on Visualization and Computer Graphics*, 31(4):2114–2126, 2025. doi: 10.1109/TVCG.2024.3381453 1, 2, 3, 4

[16] J. Venna and S. Kaski. Local Multidimensional Scaling. *Neural Networks*, 19(6–7):889–899, 2006. doi: 10.1016/j.neunet.2006.05.014 3

[17] J. Xia, Y. Zhang, J. Song, Y. Chen, Y. Wang, and S. Liu. Revisiting Dimensionality Reduction Techniques for Visual Cluster Analysis: An Empirical Study. *IEEE Transactions on Visualization and Computer Graphics*, 28(1):529–539, 2022. doi: 10.1109/TVCG.2021.3114694 1

[18] L. Zhou, C. R. Johnson, and D. Weiskopf. Data-Driven Space-Filling Curves. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1591–1600, 2021. doi: 10.1109/TVCG.2020.3030473 1